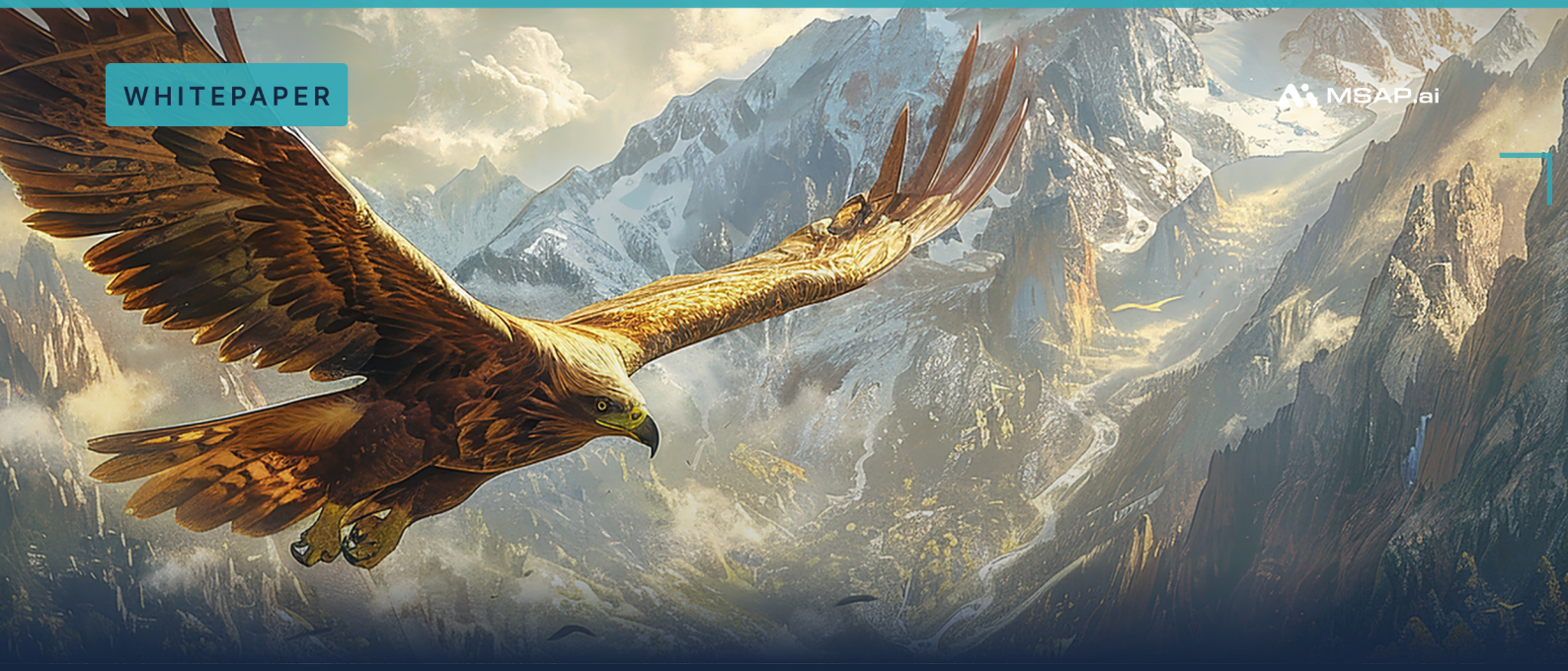


WHITEPAPER

MSAP.ai

말하는 챗봇에서 일하는 AI Agent로

업무 담당자가 만드는 OPENMARU AAS

질문에 답하는 챗봇을 넘어 업무를 끝까지 처리하는 AI Agent는 업무를 아는 담당자가 말로 정의해 만들고, 외부 LLM에 자료를 보내지 않는 조직 안의 sLLM 위에서 일하게 할 때 업무에 자리 잡습니다. 이 백서는 그 방법을 견적 작성 사례와 OPENMARU AAS로 설명합니다.

2026-11-11 · 작성자: OPENMARU · msap.ai

목차

말하는 챗봇에서 일하는 AI Agent로 — 업무 담당자가 만들고 조직 안에서 일하는 OPENMARU AAS

1장: 말하는 챗봇과 일하는 AI Agent의 차이

1.1 한 번 묻고 한 번 답하는 사내 RAG 챗봇과 업무를 끝까지 처리하는 AI Agent의 구조 차이

1.2 자율성·도구·기억·권한·실행 반복으로 가르는 챗봇과 AI Agent의 다섯 가지 기준

1.3 「사내 챗봇으로 충분하다」는 반론과 기존 챗봇 자산의 재사용 범위

1.4 챗봇으로 끝나지 않는 업무의 판별 기준 — 여러 시스템을 거치는 반복 업무

2장: AI 산업이 AI Agent 중심으로 옮겨 가는 이유

2.1 AI 정기 사용 88%와 회사 전체 확산 7% 사이의 간격

2.2 2026–2030년 업무 특화 에이전트와 에이전틱 AI 소프트웨어 시장 전망

2.3 PoC 뒤 중단 최소 30%·에이전틱 AI 프로젝트 취소 40% 이상이라는 반대편 신호

2.4 답변 품질 경쟁에서 업무 완결 경쟁으로 — 유행과 구조 변화를 가르는 판단 근거

3장: Prompt·Context·Loop·AI Agent·Agentic AI 다섯 단계 비교

3.1 사람의 개입 범위로 나눈 다섯 단계와 단계별 사람의 역할

3.2 계획·실행·확인·재시도를 스스로 반복하는 Loop 단계의 정의

3.3 한 업무를 끝까지 처리하는 AI Agent와 여러 에이전트를 조율하는 Agentic AI의 경계

3.4 업무가 요구하는 가장 낮은 단계에서 시작하는 단계 진단 원칙

4장: 외부 LLM에 자료를 보내지 않는 내부 sLLM과 폐쇄망 AI Agent

4.1 질문 한 줄에서 업무 자료 전체로 넓어지는 AI Agent의 모델 전송 범위

4.2 「sLLM은 성능이 낮다」는 반론과 판단은 코드·요청 해석은 모델이 맡는 설계

4.3 모든 모델 호출이 사내 모델 게이트웨이를 거치는 폐쇄망 AI Agent 구성

4.4 예시 모델로 본 sLLM 장비 판정과 자체 서빙 비용의 판단 방향

5장: AI Agent를 만드는 주체인 업무 담당자

5.1 AI 개발자 채용·외부 구축·업무 담당자 — 제작 주체 세 가지 비교

5.2 목적·대상·판정 기준·검수로 업무를 가르치는 에이전트 4계층 책임 구조

5.3 「업무 담당자가 정말 만들 수 있나」라는 의문과 말로 정하는 범위의 경계

5.4 부서가 직접 업무 프로그램을 만들던 전례와 담당자가 바뀌어도 남는 스킬

6장: 견적 작성 사례로 보는 자연어 업무 정의와 에이전트 코딩

6.1 만드는 단계와 일하는 단계의 분리 — 코딩 에이전트와 사내 sLLM 에이전트의 역할

6.2 공공기관 VAT 포함·조달 등록가 기준 등 업무 담당자가 말로 정한 견적 작성 규칙

6.3 「견적」 요청에서 CRM 등록·PDF 저장·영업기회 일지 기록까지의 처리 9단계

6.4 견적 작성 다음 후보 업무의 판별 기준 — 문장으로 정리되는 규칙과 반복 빈도

7장: OPENMARU AAS를 만든 배경과 제품 정의

7.1 에이전트 플랫폼의 채팅 창구가 남긴 세 가지 운영 공백

7.2 만들기·시키기·지켜보기·나누기를 한 화면에 담은 업무 담당자용 Studio의 정의

7.3 에이전트 플랫폼 대시보드와 OPENMARU AAS의 역할 구분 — 전체 관리와 사용자별 에이전트 운영

8장: AI Agent 개발 도구·빌더와 비교한 특징과 차별점
8.1 만들기 중심·지켜보기 중심과 SaaS·조직 안 설치로 나눈 에이전트 도구 여섯 유형
8.2 「빌더는 이미 많다」는 반론과 캔버스 조립·말로 정의하는 제작 방식의 차이
8.3 OPENMARU AAS가 앞서는 영역 — 조직 안 설치·사용자별 작업공간·운영 가시성
8.4 OPENMARU AAS가 아직 채우지 못한 영역 — 단일 플랫폼 연동·결제형 마켓 부재·조회 중심 비용 통제
9장: 사용자와 관리자가 쓰는 주요 기능
9.1 대화에서 승인과 예약 작업까지 이어지는 업무 담당자 화면
9.2 스킬 만들기·게시 검토·허브 설치로 이어지는 스킬 수명 주기
9.3 관리자·팀 관리자·작업공간 주인·일반 사용자 4종 권한과 작업공간 한도
9.4 화면에서 끌 수 없는 질문 가드레일·위험 명령 차단과 사용량·변경 기록 점검
10장: OPENMARU AAS가 있을 때와 없을 때의 업무 비교
10.1 업무 장면 8가지로 본 OPENMARU AAS가 있을 때와 없을 때
10.2 업무 담당자 쪽 변화 — 규칙 재설명과 결과 재입력 수작업의 축소
10.3 IT 담당자와 조직 쪽 변화 — 흩어진 에이전트·자격증명·예약 작업의 한 화면 관리
10.4 시범 도입 효과를 결재 문서에 적는 지표 — 처리 시간·결과 품질·운영 통제
11장: 도입 조직과 개발자 입장에서 본 Apache-2.0 오픈소스
11.1 코드는 Apache-2.0, 이름·로고는 상표 정책 — 도입 조직의 설치·수정·재배포 권리
11.2 OSI 승인 라이선스 여부가 가르는 공급사 종속과 도입 위험
11.3 Docker Compose 공개본과 비공개 쿠버네티스 구성의 기능 경계
11.4 「오픈소스면 지원은 누가 하나」라는 질문과 공개 코어·조직 스킬·사내 허브의 책임 구분
12장: 단계별 발전 방향과 도입 순서
12.1 발전 8단계에서 본 OPENMARU AAS의 현재 위치와 다음 순서
12.2 AI 단계 진단·sLLM 준비·첫 업무 정의·부서 확산의 도입 4단계
12.3 시범 도입 대상이 될 첫 업무의 선정 기준 — 말로 정할 수 있는 규칙·반복 빈도·조직 안에 둘 자료
12.4 AI 단계·내부 sLLM 필요성·제작 주체·첫 업무 — 의사결정권자를 위한 도입 결재 점검표
부록 A. 용어 풀이
부록 B. 참고문헌

말하는 챗봇에서 일하는 AI Agent로 — 업무 담당자가 만들고 조직 안에서 일하는 OPENMARU AAS

이 백서는 사내 챗봇 다음 단계로 AI Agent 도입을 검토하는 의사결정권자와 IT 담당자를 위한 자료입니다. 1–4장은 챗봇과 AI Agent의 차이, AI 산업이 AI Agent로 옮겨 가는 이유, 조직 AI의 다섯 단계, 내부 sLLM과 폐쇄망이 필요한 이유를 차례로 짚습니다. 5–7장은 업무 담당자가 AI Agent를 만드는 주체가 되는 방식을 견적 작성 사례와 OPENMARU AAS의 제품 정의로 설명하고, 8–11장은 다른 에이전트 도구와의 비교, 주요 기능, 있을 때와 없을 때의 업무 차이, Apache-2.0 오픈소스의 의미를 다룹니다. 12장은 단계별 발전 방향과 도입 순서를 거쳐, 첫 시범 도입 업무를 고르는 결재 점검표로 이어집니다.

1장: 말하는 챗봇과 일하는 AI Agent의 차이

많은 조직이 사내 자료를 검색해 답을 만들어 주는 RAG(Retrieval-Augmented Generation) 챗봇을 이미 운영하고 있습니다. RAG 챗봇은 질문을 받으면 사내 자료에서 근거를 찾아 답변 한 건을 돌려주고, 그 답변을 업무 시스템에 반영하는 일은 여전히 사람이 맡습니다. AI Agent(AI 에이전트)는 같은 언어 모델을 쓰면서도 도구·기억·권한을 함께 갖추고 업무를 끝까지 처리해 보고서·견적서·처리 기록 같은 결과물을 남기는 시스템입니다. 「사내 챗봇을 이미 도입했는데 AI Agent는 무엇이 다른가」라는 질문에는 답변까지 처리하는 시스템과 결과물까지 처리하는 시스템의 차이로 답할 수 있습니다. 그 차이는 다섯 가지 기준으로 정리할 수 있고, 같은 기준으로 기존 챗봇 투자를 AI Agent에서 어디까지 이어 쓰는지와 챗봇으로 끝나지 않는 업무를 어떻게 고르는지도 판단할 수 있습니다.

1.1 한 번 묻고 한 번 답하는 사내 RAG 챗봇과 업무를 끝까지 처리하는 AI Agent의 구조 차이

사내 RAG 챗봇과 AI Agent는 둘 다 대화창에서 요청을 받기 때문에 겉으로는 비슷해 보입니다. 하지만 RAG 챗봇은 질문 한 번에 응답 한 번으로 처리가 끝나는 단일 턴 구조이고, AI Agent는 목표를 받은 뒤 계획을 세우고 도구를 호출하고 결과를 확인하는 과정을 업무가 끝날 때까지 반복하는 구조입니다[S11]. 두 구조의 차이는 처리 범위에서 드러납니다. 챗봇의 처리 범위는 답변까지이고, AI Agent의 처리 범위는 업무 시스템에 남는 결과물까지입니다.

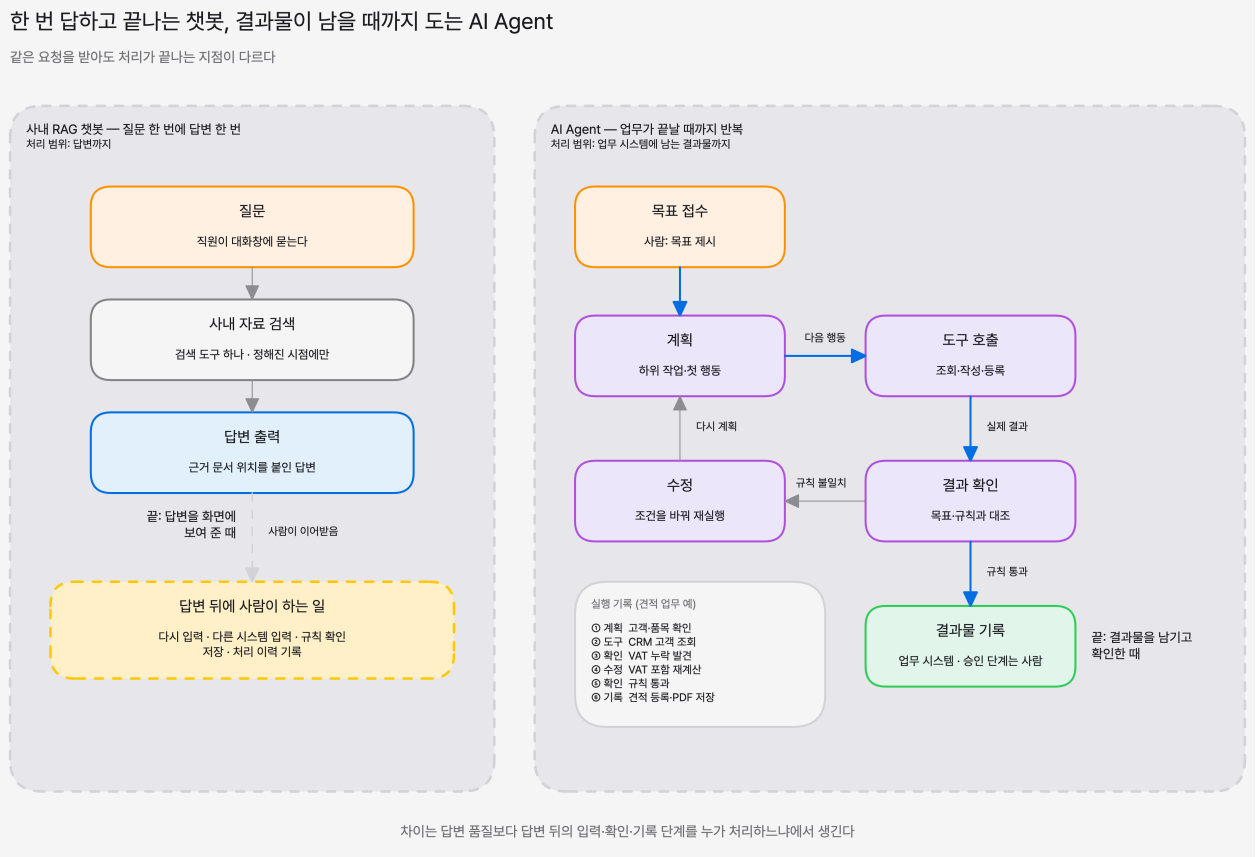


그림. 질문→사내 자료 검색→답변 출력으로 끝나는 RAG 챗봇의 단일 턴과, 계획→도구 호출→결과 확인→수정을 반복해 업무 시스템에 결과물을 남기는 AI Agent의 실행 반복

구분	사내 RAG 챗봇	AI Agent
처리 단위	질문 한 건과 답변 한 건	업무 한 건의 시작부터 결과물 저장까지
끝나는 시점	답변을 화면에 보여 준 때	결과물을 업무 시스템에 남기고 확인한 때
사람이 맡는 일	답변을 다른 시스템에 다시 입력하고 확인	목표 제시와 승인이 필요한 단계의 확인

1.1.1 사내 문서 검색과 요약에 머무는 RAG 챗봇의 처리 범위

RAG 챗봇이 잘하는 사내 자료 검색

RAG 챗봇은 사내 규정, 제품 설명서, 과거 보고서처럼 흩어진 문서에서 질문과 관련된 단락을 찾아 언어 모델에게 함께 넘기고, 모델은 그 근거를 바탕으로 답변을 만듭니다. 덕분에 일반 언어 모델이 모르는 사내 용어와 규정에도 근거를 붙여 답할 수 있고, 담당자가 문서함을 직접 뒤지는 시간이 줄어듭니다. 「출장비 정산 한도가 얼마인가」 같은 규정 문의, 「이 제품의 지원 운영체제는 무엇인가」 같은 사양 문의는 RAG 챗봇이 가장 잘 처리하는 질문입니다. 이러한 성과는 사내 자료를 정리하고 검색 색인을 만든 투자에서 나온 것이며, 다음 단계로 넘어가도 그 가치는 그대로 남습니다.

검색 도구 하나에 고정된 RAG 챗봇의 경계

다만 RAG 챗봇이 쓰는 도구는 대개 문서 검색 하나이고, 언제 검색할지도 설계 시점에 정해져 있습니다. RAG 챗봇은 「도구를 하나 가진 챗봇」에 가깝고, 질문을 받을 때마다 검색하고 답하는 단일 턴 구조에서 벗어나지 않습니다. 이 구조에서는 답변에 이어 업무 시스템을 조회하거나, 서식을 채워 문서를 만들거나, 처리 결과를 기록하는 단계가 시스템 밖에 남습니다. 결국 RAG 챗봇의 처리 범위는 사내 자료를 찾아 요약한 답변까지이고, 그다음 단계는 처음부터 설계 범위에 들어 있지 않습니다.

처리 범위로 정리한 RAG 챗봇의 역할

RAG 챗봇이 처리하는 범위는 다음 네 가지입니다.

- 사내 자료에서 질문과 관련된 근거 찾기
- 찾은 근거를 요약해 답변 문장 만들기
- 답변에 근거 문서의 위치 붙이기
- 같은 대화창 안에서 이어지는 후속 질문에 답하기

이 목록에는 업무 시스템에 쓰는 동작이 하나도 없습니다. RAG 챗봇은 업무 담당자의 판단을 돕는 참고 창구로는 충분하지만, 업무를 맡아서 결과물을 남기는 역할은 처음부터 설계 범위 밖에 있습니다.

1.1.2 답변 뒤에 사람이 떠안는 재입력·시스템 입력·확인 작업

답변 이후에 남는 수작업 단계

챗봇이 좋은 답변을 내놓아도 업무는 그 답변에서 끝나지 않는 경우가 많습니다. 예를 들어 견적 문의에 대한 답변을 받은 담당자는 답변 내용을 영업 관리 시스템에 다시 입력하고, 서식에 맞춰 견적서를 만들고, 할인 규칙과 세금 계산이 맞는지 확인하고, 처리 결과를 고객 기록에 남깁니다. 이 단계들은 모두 사람이 직접 처리하므로, 챗봇이 줄여 준 시간은 검색과 초안 작성에 쓰던 시간에 한정됩니다. 업무 전체 처리 시간에서 챗봇이 줄일 수 있는 몫은 검색 비중이 큰 업무일수록 크고, 시스템 입력과 확인 비중이 큰 업무일수록 작습니다.

답변 이후 수작업 단계의 목록

답변을 받은 뒤 사람이 처리하는 작업은 업무마다 다르지만, 대체로 다음 다섯 단계로 정리됩니다.

1. 답변 내용을 업무 시스템 입력 양식에 맞게 다시 쓰기
2. 영업 관리·결재·문서 관리 같은 다른 시스템에 직접 입력하기
3. 계산 결과와 사내 규칙 위반 여부를 눈으로 확인하기
4. 결과 문서를 정해진 저장소에 올리고 관련자에게 알리기
5. 처리 이력을 일지나 고객 기록에 남기기

수작업 단계가 만드는 일관성 문제

이 다섯 단계는 사람이 처리하기 때문에 담당자마다 방식이 조금씩 다르고, 바쁜 날에는 확인 단계가 빠지기도 합니다. 반면 같은 단계를 AI Agent가 업무 규칙에 따라 처리하면 입력 양식, 계산 방식, 확인 항목이 매번 같은 기준으로 적용되고 처리 기록도 남습니다. 챗봇과 AI Agent의 차이는 답변 품질보다 답변 이후 단계를 누가 처리하느냐에서 더 크게 나타납니다. 의사결정권자가 먼저 확인할 사항은 지금 운영하는 챗봇의 답변 뒤에 이 다섯 단계가 얼마나 남아 있는지입니다.

사람이 보지 않는 실행이 요구하는 실행 기록

다만 AI Agent에게 이 단계를 맡기면 사람이 중간 결과를 읽고 다음 지시를 내리던 확인 절차도 함께 줄어듭니다. 챗봇은 질문 하나에 답 하나를 돌려주고 멈추므로 다음 행동을 정하는 쪽이 늘 사람입니다. 반면 AI Agent는 어떤 도구를 부르고 언제 멈출지를 스스로 정하며 여러 단계를 이어 갑니다. 그래서 한 단계의 잘못된 판단이 그대로 다음 단계의 입력이 될 수 있습니다. 예를 들어 조회 단계에서 기간을 잘못 잡으면 작성 단계는 틀린 자료로 그럴듯한 보고서를 만들고, 발송 단계는 그 보고서를 그대로 내보냅니다. AI Agent의 실행 기록은 성공과 실패 여부만이 아니라 각 실행이 어떤 스킬과 도구를 거쳤는지까지 보여 주어야 합니다. 그래야 결과가 이상할 때 담당자가 원인을 거꾸로 따라갈 수 있습니다.

1.2 자율성·도구·기억·권한·실행 반복으로 가르는 챗봇과 AI Agent의 다섯 가지 기준

챗봇과 AI Agent를 구분하는 기준은 자율성, 도구, 기억, 권한, 실행 반복의 다섯 가지이며, 이 백서의 이후 장도 같은 기준과 용어를 따릅니다. 다섯 기준은 둘 중 하나를 고르는 구분보다 정도의 차이에 가깝기 때문에, 한 시스템이 기준마다 다른 수준에 있을 수 있습니다. RAG 챗봇은 도구 기준에서는 검색 도구를 갖추어 AI Agent 쪽으로 한 걸음 나아갔지만, 실행 반복과 권한 기준에서는 일반 챗봇과 같은 수준입니다.

기준	챗봇(RAG 챗봇 포함)	AI Agent
자율성	사람이 질문할 때마다 정해진 방식으로 한 번 응답	목표를 받아 다음에 할 행동을 스스로 정함
도구	없거나 문서 검색 하나로 고정	조회·작성·등록 도구를 상황에 맞게 골라 호출
기억	현재 대화창 안의 맥락만 유지	업무 규칙·작업 이력·중간 결과를 저장해 다음 실행에서 참조
권한	검색 대상 문서를 읽는 권한	업무 시스템에 쓰는 권한, 승인이 필요한 행동은 사람 확인
실행 반복	질문 한 번에 응답 한 번	계획→실행→결과 확인→수정을 업무가 끝날 때까지 반복

1.2.1 정해진 코드 경로의 워크플로와 스스로 경로를 정하는 에이전트의 구분

Anthropic이 정리한 워크플로와 에이전트의 정의

언어 모델과 도구를 엮은 시스템은 실행 경로를 누가 정하느냐에 따라 워크플로와 에이전트로 나뉩니다. Anthropic은 2024년 12월 공개한 글에서 두 방식을 다음과 같이 정의했습니다[S11].

> *"Workflows are systems where LLMs and tools are orchestrated through predefined code paths." / "Agents … are systems where LLMs dynamically direct their own processes and tool usage"*

워크플로는 사람이 미리 작성한 코드 경로를 따라 언어 모델과 도구가 정해진 순서로 실행되는 시스템이고, 에이전트는 언어 모델이 실행 중에 자신의 처리 과정과 도구 사용을 직접 정하는 시스템입니다[S11]. 두 방식은 우열의 문제라기보다 업무 성격에 따른 선택의 문제이며, 같은 글은 가능한 한 가장 단순한 해법에서 시작하고 필요할 때만 복잡도를 높이라고 권합니다[S11].

업무 성격에 따른 워크플로와 에이전트의 선택

순서가 늘 같은 업무는 워크플로로 처리하는 편이 예측하기 쉽고 비용도 낮습니다. 반면 요청마다 조회할 시스템과 확인할 규칙이 달라지는 업무는 미리 모든 경로를 코드로 그려 두기 어렵기 때문에, 언어 모델이 상황을 보고 다음 행동을 정하는 에이전트 방식이 맞습니다. 공공기관 견적과 민간 견적은 세금 표시와 할인 규칙이 다르고, 조달 품목이 섞이면 단가 계산 방식도 달라집니다. 이런 업무에서는 경로 판단을 에이전트에게 맡기되, 판단 기준이 되는 업무 규칙은 사람이 정해 두는 구성이 현실적입니다.

이 백서에서 쓰는 AI Agent의 범위

이 백서는 언어 모델이 도구·기억·권한을 갖추고 실행 경로를 스스로 정하면서 업무 한 건을 결과물까지 처리하는 시스템을 AI Agent로 부릅니다. 여러 AI Agent가 역할을 나눠 서로 조율하며 일하는 구성은 Agentic AI로 따로 부릅니다. 정의를 이렇게 고정해 두면 3장에서 Prompt, Context, Loop, AI Agent, Agentic AI 다섯 단계를 비교할 때 단계 사이의 경계가 흔들리지 않습니다.

1.2.2 추론과 실행을 번갈아 수행하는 ReAct 구조와 환경 피드백 반복

ReAct 논문이 제시한 추론과 실행의 교차

AI Agent의 실행 반복을 처음 체계적으로 보인 연구로는 Yao 등이 2022년 10월 공개하고 ICLR 2023에서 발표한 ReAct(Reasoning and Acting)가 자주 꼽힙니다[S10]. ReAct는 언어 모델이 추론 과정과 업무별 행동을 번갈아 생성하도록 구성한 방식이며, 원문 초록은 이를 "reasoning traces and task-specific actions in an interleaved manner" 로 설명합니다[S10]. 이 구조에서 추론 과정은 행동 계획을 세우고, 진행 상황을 추적하고, 계획을 고치고, 예외를 처리하는 데 쓰입니다[S10]. 모델은 한 번에 정답을 내놓는 대신, 생각하고 행동하고 결과를 본 뒤 다시 생각하는 순서로 문제를 풉니다.

환경 피드백을 받아 반복하는 실행 구조

Anthropic은 같은 구조를 실무 관점에서 더 짧게 정리했습니다. 에이전트는 대개 환경에서 돌아오는 피드백을 보며 도구를 쓰는 반복 구조로 동작하는 언어 모델이고, 매 단계에서 도구 호출 결과나 코드 실행 결과 같은 실제 결과를 받아 진행 상황을 판단해야 합니다[S11]. 여기서 말하는 실제 결과는 업무 시스템이 돌려준 조회 결과, 계산 결과, 오류 메시지 같은 것입니다. 따라서 AI Agent의 신뢰성은 모델이 얼마나 그럴듯하게 말하느냐보다 매 단계의 실제 결과를 확인하고 틀리면 고치는 반복을 갖추었느냐에 따라 정해집니다.

계획에서 수정까지 이어지는 네 단계 반복

업무 한 건을 처리하는 AI Agent의 실행 반복은 다음 네 단계로 정리됩니다.

1. 계획 — 목표를 하위 작업으로 나누고 첫 행동을 정한다
2. 실행 — 조회·작성·등록 도구를 호출한다
3. 결과 확인 — 도구가 돌려준 결과가 목표와 규칙에 맞는지 본다
4. 수정 — 맞지 않으면 조건을 바꿔 다시 실행하고, 맞으면 다음 하위 작업으로 넘어간다

이 반복은 3장에서 Loop 단계로 다시 정의하며, AI Agent는 이 반복에 도구·기억·권한을 더해 업무 한 건을 결과물까지 처리합니다.

1.3 「사내 챗봇으로 충분하다」는 반론과 기존 챗봇 자산의 재사용 범위

「우리는 이미 사내 챗봇이 있으니 충분하다」는 반론은 AI Agent 검토 초기에 가장 자주 나오는 의견이며, 질의응답이 주된 업무라면 맞는 판단입니다. 업무가 답변 이후의 시스템 입력과 확인으로 이어진다면 챗봇만으로는 처리 범위가 부족하고, 이때도 기존 챗봇 투자는 버려지지 않고 AI Agent의 부품으로 이어집니다. 검색 색인과 문서 연결은 AI Agent가 쓰는 검색 도구가 되고, 직원이 익숙하게 쓰는 질의 화면은 AI Agent에게 일을 맡기는 창구로 이어집니다.

1.3.1 사내 자료 연결과 질의 창구로 이어 쓰는 기존 챗봇 투자

검색 색인과 문서 연결의 재사용

RAG 챗봇을 만들면서 조직이 갖춘 가장 큰 자산은 정리된 사내 자료, 검색 색인, 문서별 열람 권한입니다. AI Agent는 업무를 처리하는 중에 규정이나 과거 사례를 찾아야 할 때 이 검색 기능을 하나의 도구로 호출하므로, 색인과 문서 연결을 새로 만들 필요가 없습니다. 예를 들어 OPENMARU AAS에서는 외부 기능을 MCP(Model Context Protocol, 언어 모델이 외부 도구와 데이터에 접속하는 표준 규약) 서버로 연결해 에이전트의 도구로 등록할 수 있으므로, 기존 검색 서비스를 그런 방식으로 이어 붙일 수 있습니다. 챗봇 구축에 들인 데이터 정리 비용은 AI Agent 단계에서도 그대로 회수됩니다.

AI Agent를 위해 새로 갖출 항목

업무 시스템에 쓰는 연결, 업무 규칙, 승인 절차, 실행 기록은 대부분의 챗봇에 없던 항목이라 새로 갖춰야 합니다. 도입 검토는 기존 자산 목록과 새로 필요한 항목 목록을 나란히 놓고 시작하는 편이 정확합니다.

챗봇 자산·기능	AI Agent 에서의 쓰임	판단
정리된 사내 자료와 검색 색인	규정·사례를 찾는 검색 도구	재사용 가능
문서별 열람 권한	에이전트가 볼 수 있는 자료 범위	재사용 가능
질의 화면과 사용자 계정	일을 맡기고 결과를 받는 창구	재사용 가능(연결 방식 확인 필요)
업무 시스템 쓰기 연결	조회·등록·문서 생성 도구	추가 필요
업무 규칙과 점검 기준	결과물이 맞는지 판정하는 기준	추가 필요
승인 절차와 실행 기록	사람 확인 단계와 처리 이력	추가 필요

이 표에서 「추가 필요」로 분류한 항목은 업무 담당자가 규칙을 정하고 IT 담당자가 연결과 권한을 정하는 영역이며, 5장과 6장에서 그 방법을 다룹니다.

1.3.2 민원 처리 한 건으로 비교한 답변형 AI와 일하는 AI Agent의 처리 범위

같은 민원 요청에 대한 두 방식의 처리 단계

공공기관의 민원 처리 한 건을 예로 들면 답변형 AI와 일하는 AI Agent의 처리 범위가 단계별로 드러납니다. 민원이 접수되면 담당자는 관련 규정을 찾고, 민원인의 과거 이력을 조회하고, 답변 초안을 쓰고, 결재 시스템에 올리고, 처리 결과를 기록합니다. RAG 챗봇은 이 가운데 규정 검색과 답변 초안 작성을 돕지만, 나머지 단계는 담당자가 직접 처리합니다. 반면 AI Agent는 이력 조회, 결재 초안 등록, 처리 기록까지 처리하고 담당자는 결재 단계에서 내용을 확인하고 승인합니다.

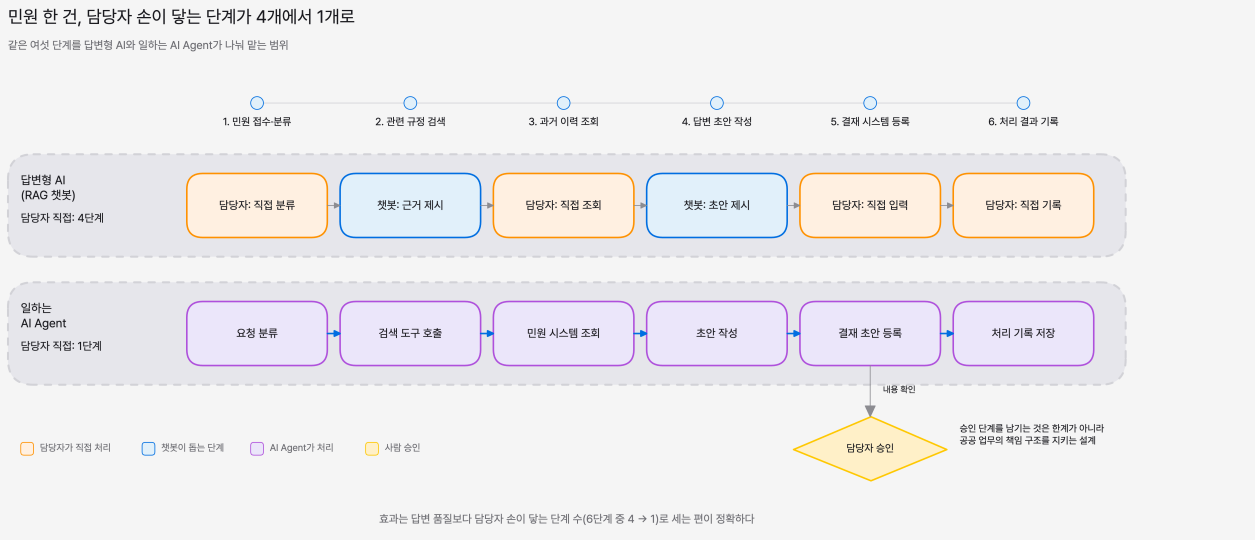


그림. 민원 한 건의 여섯 단계를 RAG 챗봇과 AI Agent가 나눠 맡는 범위 — 담당자 손이 닿는 단계가 4개에서 결재 승인 1개로 줄어든다

민원 처리 단계	답변형 AI(RAG 챗봇)	일하는 AI Agent
민원 접수와 분류	담당자	AI Agent
관련 규정 검색	챗봇이 근거 제시	AI Agent(검색 도구 호출)
민원인 과거 이력 조회	담당자	AI Agent(민원 시스템 조회)
답변 초안 작성	챗봇이 초안 제시	AI Agent
결재 시스템 등록	담당자	AI Agent가 초안 등록, 담당자가 승인
처리 결과 기록	담당자	AI Agent

두 방식의 차이가 결과물에서 드러나는 이유

표에서 보듯 두 방식 모두 규정 검색과 초안 작성에는 언어 모델을 쓰지만, 결과물이 업무 시스템에 남는지는 다릅니다. 챗봇 방식에서는 담당자가 여섯 단계 가운데 네 단계를 직접 처리하고, AI Agent 방식에서는 담당자가 승인 한 단계에 집중합니다. 승인 단계를 남겨 두는 것은 AI Agent의 한계라기보다 공공 업무의 책임 구조를 지키는 설계입니다. AI Agent 도입 효과를 따질 때는 담당자의 손이 닿는 단계가 몇 개에서 몇 개로 줄어드는지를 세는 편이 답변 품질을 비교하는 것보다 정확합니다.

1.4 챗봇으로 끝나지 않는 업무의 판별 기준 — 여러 시스템을 거치는 반복 업무

챗봇으로 끝나지 않는 업무는 대개 두 개 이상의 시스템을 거치고, 같은 절차가 자주 반복되며, 사람이 확인해야 하는 단계가 분명한 업무입니다. 이 세 가지 성격을 갖춘 업무는 단일 턴 답변으로 처리가 끝나지 않기 때문에, 챗봇을 더 고도화해도 담당자의 수작업이 크게 줄지 않습니다. 반대로 세 가지 가운데 어느 것도 해당하지 않는 업무는 지금의 챗봇으로 충분하며, 가장 단순한 해법부터 쓰라는 원칙에도 맞습니다[S11].

1.4.1 연결 시스템 수·반복 빈도·사람 확인 지점으로 구성한 판별 체크리스트

판별 체크리스트의 세 항목

판별 기준은 연결 시스템 수, 반복 빈도, 사람 확인 단계의 세 항목으로 구성합니다. 연결 시스템 수는 업무 한 건을 끝내려고 열어야 하는 시스템의 개수이고, 반복 빈도는 같은 절차가 얼마나 자주 되풀이되는지를 뜻합니다. 사람 확인 단계는 담당자가 반드시 보고 승인해야 하는 단계가 어디인지 분명한지를 봅니다.

항목	해당 기준	예시 업무
연결 시스템 수	업무 한 건에 두 개 이상의 시스템을 조회하거나 입력한다	고객 관리 시스템 조회 후 견적서 작성과 문서 저장
반복 빈도	같은 절차를 주 1회 이상 되풀이하고 절차가 문장으로 설명된다	주간 실적 보고서, 정기 점검 결과 정리
사람 확인 단계	승인·검토해야 할 단계가 한두 곳으로 정해져 있다	견적 발송 전 금액 확인, 결재 상신 전 내용 확인

체크리스트 세 항목의 종합 판정

세 항목에 모두 해당하는 업무가 AI Agent의 첫 후보입니다. 두 항목만 해당하면 업무 범위를 조정해 다시 판정하고, 한 항목 이하이면 챗봇이나 단순 자동화로 충분한 경우가 많습니다. 특히 사람 확인 단계가 정해지지 않은 업무는 AI Agent에게 맡겨도 결과를 누가 언제 책임지는지가 불분명해지므로, 확인 단계를 먼저 정한 뒤에 후보로 올립니다. 이 체크리스트는 도입 대상을 넓히는 도구보다 첫 업무 하나를 좁혀 고르는 도구로 쓰는 편이 효과적입니다.

결과를 확인할 사람을 먼저 적어 두는 이유

AI Agent가 단계마다 사람에게 묻지 않는다는 점은 처리 속도를 높이는 장점이면서 사람의 확인이 필요한 이유이기도 합니다. 첫 단계에서 잘못 고른 조건이 이후 단계에 그대로 이어질 수 있고, 도구 실행을 되풀이하는 만큼 모델이 읽고 쓰는 글의 양과 비용도 늘어납니다. 체크리스트의 세 번째 항목에는 확인 단계의 위치와 함께 그 단계에서 결과를 확인할 사람까지 적어 두는 것이 좋습니다. AI Agent를 도입해도 기존 업무 시스템과 사람의 역할은 그대로 남고, 바뀌는 것은 반복 업무를 처리하는 주체입니다. 결과를 판단할 사람이 처음부터 정해져 있어야 에이전트가 확인 단계에 이르거나 진행이 막혔을 때 일이 멈추지 않고 그 사람에게 넘어갑니다.

1.4.2 결과물을 남기는 업무와 답변으로 끝나는 업무의 구분 사례

결과물이 남는 업무와 답변으로 끝나는 업무의 대비

판별 체크리스트를 실제 업무에 적용하면 업무는 결과물이 남는 업무와 답변으로 끝나는 업무로 나뉩니다. 결과물이 남는 업무는 보고서, 견적서, 업무 일지처럼 문서나 기록이 업무 시스템에 저장되고 다른 사람이 그 결과물을 이어받아 씁니다. 답변으로 끝나는 업무는 질문한 사람이 답을 읽고 판단하면 처리가 끝나며, 별도로 저장할 결과물이 없습니다.

결과물이 남는 업무	답변으로 끝나는 업무
주간 실적 보고서 작성과 공유	사내 규정·한도 문의
고객 요청에 맞춘 견적서 작성과 저장	제품 사양·지원 범위 문의

결과물이 남는 업무	답변으로 끝나는 업무
영업 활동 일지 기록	용어 설명과 문서 위치 찾기
정기 점검 결과 정리와 담당자 알림	과거 결정 사항의 근거 확인

구분 결과를 도입 판단에 쓰는 방법

왼쪽 열의 업무는 AI Agent가 처리하면 결과물이 업무 시스템에 남고 처리 기록도 함께 쌓이므로, 도입 효과를 처리 건수와 담당자 확인 시간으로 셀 수 있습니다. 오른쪽 열의 업무는 지금의 RAG 챗봇이 이미 잘 처리하는 영역이므로 그대로 두는 편이 비용 면에서 합리적입니다. 견적서 작성은 고객 관리 시스템 조회, 금액 계산, 규칙 점검, 문서 저장을 모두 거치는 대표적인 결과물 업무이며, 6장에서 이 업무를 사례로 업무 담당자가 말로 규칙을 정의하는 과정을 설명합니다. 따라서 의사결정권자가 1장에서 내릴 판단은 우리 조직에 왼쪽 열에 해당하는 반복 업무가 있는지, 있다면 그중 확인 단계가 가장 분명한 업무가 무엇인지입니다.

2장: AI 산업이 AI Agent 중심으로 옮겨 가는 이유

1장에서 가른 답변과 결과물의 차이는 산업 전체의 수치에서도 드러납니다. McKinsey가 2025년 11월 발표한 「The state of AI in 2025」에서 응답 조직의 88%는 한 개 이상의 업무 기능에서 AI를 정기적으로 사용한다고 답했지만, AI를 회사 전체로 완전히 확산했다고 답한 비율은 7%에 그쳤습니다[S1][S2]. 같은 시기 Gartner는 2026년 말까지 기업용 애플리케이션의 40%가 업무 특화 AI 에이전트를 통합할 것으로 전망했고, 동시에 2027년 말까지 에이전틱 AI 프로젝트의 40% 이상이 취소될 것이라는 경고도 내놓았습니다[S7][S4]. 확산 전망과 실패 전망이 함께 나오는 지금, 의사결정권자에게 필요한 것은 AI Agent가 유행인지 구조 변화인지를 가르는 판단 근거입니다. 그 근거는 도입률과 확산률 사이의 간격, 2026–2030년 전망, 반대편 실패 신호를 함께 놓고 볼 때 드러나며, 구조 변화 여부는 결국 AI Agent가 업무에 정착했는가로 판정할 수 있습니다.

2.1 AI 정기 사용 88%와 회사 전체 확산 7% 사이의 간격

McKinsey의 2025년 조사는 2025년 6월 25일부터 7월 29일까지 1,993명에게 받은 응답으로, AI를 쓰는 조직은 크게 늘었지만 업무 전체에 정착한 조직은 아직 소수라는 사실을 보여 줍니다. 정기 사용 88%와 생성형 AI 정기 사용 79%는 거의 모든 조직이 AI를 써 보고 있다는 뜻이고, 확산을 시작한 약 3분의 1과 완전 확산 7%는 그 사용이 업무 구조로 이어진 조직이 적다는 뜻입니다[S1][S2]. 이 간격은 많은 조직이 AI를 개인의 보조 도구로 쓰는 단계에 머물러 있음을 보여 주며, 1장에서 본 「답변 뒤에 남는 수작업」과 같은 현상을 조직 단위에서 확인한 결과이기도 합니다.

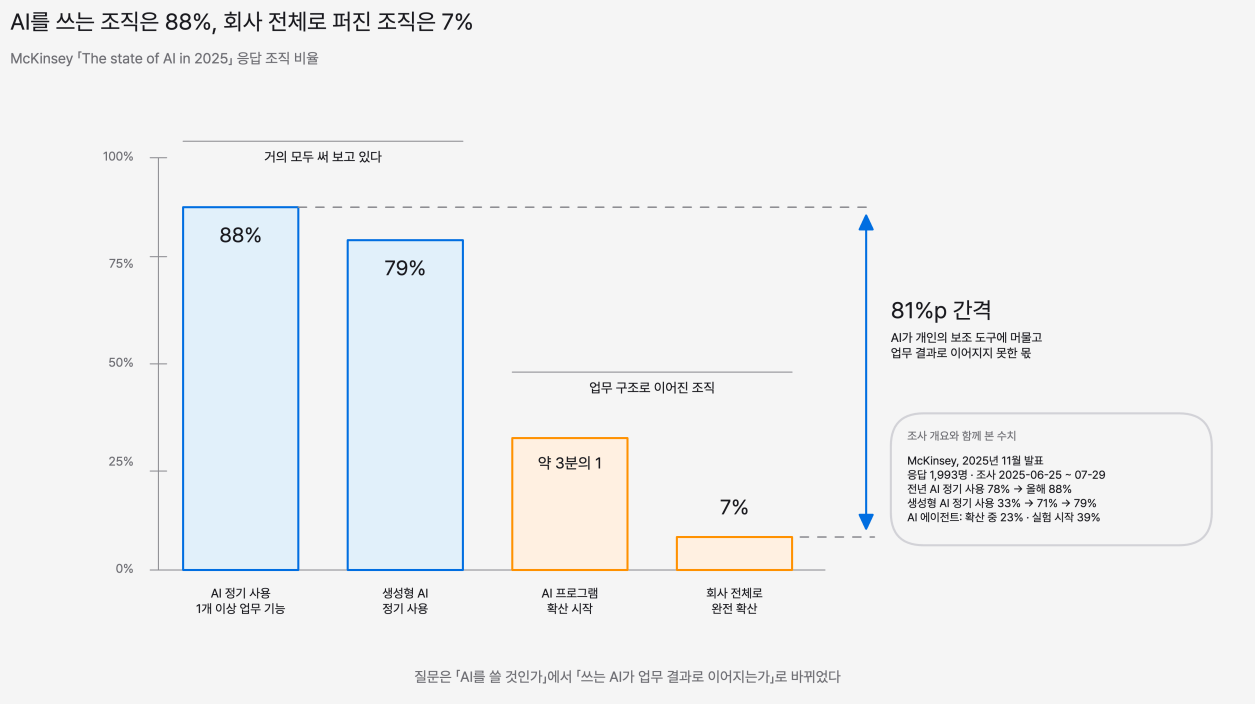


그림. AI 정기 사용 88%와 회사 전체 완전 확산 7% 사이의 간격 (McKinsey, The state of AI in 2025, 2025년 11월)

2.1.1 생성형 AI 정기 사용률 2023년 33%에서 2025년 79%까지의 추이

생성형 AI 정기 사용률의 3년 추이

생성형 AI를 정기적으로 사용하는 조직의 비율은 McKinsey 조사에서 2023년 33%, 2024년 71%, 2025년 79%로 늘었습니다[S2]. 첫해에서 둘째 해 사이에 두 배 이상 뛰었고, 셋째 해에는 증가 폭이 8%p로 줄어 사용률이 높은 수준에서 안정되는 모습입니다. 이 3개년 수치는 McKinsey가 차트로 공개한 값이므로, 다른 문서에 인용할 때는 원문 차트를 함께 확인하기를 권합니다[S2].

연도	생성형 AI 정기 사용 조직 비율	출처
2023년	33%	McKinsey, 2025년 11월
2024년	71%	McKinsey, 2025년 11월
2025년	79%	McKinsey, 2025년 11월

AI 전체 사용률 88%와 전년 78%의 비교

생성형 AI에 한정하지 않은 AI 전체 사용률은 2025년 조사에서 88%로, 전년 조사의 78%보다 10%p 높아졌습니다[S1]. 생성형 AI 사용률과 AI 전체 사용률이 모두 80% 안팎에 이르렀다는 것은 「AI를 써 볼 것인가」라는 질문이 대부분의 조직에서 이미 답을 얻었다는 뜻입니다. 따라서 지금 의사결정권자에게 남은 질문은 사용 여부보다 그 사용이 업무 결과로 이어지고 있는지입니다.

2.1.2 확산을 시작한 약 3분의 1 조직과 에이전트 확산 23%·실험 39%

AI 프로그램 확산 단계에 들어선 조직의 비율

같은 McKinsey 조사에서 대다수 조직은 아직 실험이나 시범 적용 단계에 있으며, AI 프로그램의 확산을 시작한 조직은 약 3분의 1에 머물렀습니다. 이 약 3분의 1에는 확산을 진행 중인 조직과 회사 전체로 완전히 확산한 7%가 함께 들어 있습니다[S1][S2]. 사용률 88%와 확산 시작 약 3분의 1 사이의 차이는 AI를 쓰는 조직 가운데 절반 이상이 아직 개별 부서나 개인 수준에서 AI를 활용하고 있다는 뜻으로 읽을 수 있습니다.

AI 에이전트 확산 23%와 실험 39%

AI 에이전트만 따로 보면 응답자의 23%가 조직 어딘가에서 에이전틱 AI 시스템을 확산 중이라고 답했고, 39%는 AI 에이전트 실험을 시작했다고 답했습니다[S1]. 실험을 시작한 비율이 확산 중인 비율보다 크다는 점에서, AI 에이전트는 많은 조직이 가능성을 확인하는 초기 단계에 있습니다.

응답 항목(McKinsey, 2025년 11월)	비율
한 개 이상 업무 기능에서 AI 정기 사용	88%
AI 프로그램 확산 시작	약 3분의 1
회사 전체로 완전 확산	7%
에이전틱 AI 시스템 확산 중	23%
AI 에이전트 실험 시작	39%

이 수치는 지금 AI Agent를 검토하는 조직이 뒤처진 것이 아니라 대부분의 조직과 같은 출발선에 있다는 뜻이기도 합니다.

2.2 2026–2030년 업무 특화 에이전트와 에이전틱 AI 소프트웨어 시장 전망

Gartner와 Omdia가 2024년 10월부터 2025년 9월 사이에 내놓은 전망 네 가지는 시점과 측정 대상이 서로 다르지만 모두 같은 방향을 가리킵니다. 기업용 소프트웨어 안에 업무 특화 에이전트가 들어가고, 일상 업무 결정의 일부를 에이전트가 처리하며, 이를 위한 소프트웨어 시장이 빠르게 커진다는 방향입니다[S5][S6][S7][S8]. 연구기관 한 곳의 전망이라면 유행으로 볼 여지가 있지만, 서로 다른 지표가 같은 방향을 가리킨다는 점은 시장 구조의 변화를 뒷받침하는 근거가 됩니다.

전망 시점	전망 내용	기준 값	발표 기관·날짜
2026년 말	기업용 애플리케이션의 40%가 업무 특화 AI 에이전트 통합	현재 5% 미만	Gartner, 2025-08-26
2028년	기업용 소프트웨어 애플리케이션의 33%가 에이전틱 AI 포함	2024년 1% 미만	Gartner, 2024-10-21
2028년	일상 업무 결정의 최소 15%를 에이전틱 AI가 자율 처리	2024년 0%	Gartner, 2024-10-21
2030년	기업용 에이전틱 AI 소프트웨어 시장 418억 달러	2025년 약 15억 달러	Omdia, 2025-09-22

2.2.1 2026년 말 기업 앱 40%의 업무 특화 에이전트 통합 전망

업무 특화 AI 에이전트 통합 비율 40%

Gartner는 2025년 8월 26일, 2026년 말까지 기업용 애플리케이션의 40%가 업무 특화 AI 에이전트를 통합할 것으로 전망했습니다. 발표 당시 이 비율은 5% 미만이었으므로, 약 1년 반 사이에 여덟 배 이상 늘어난다는 전망입니다[S7]. 여기서 업무 특화 에이전트는 범용 대화 비서와 달리 특정 업무 한 가지를 맡아 처리하도록 만든 에이전트를 가리킵니다.

업무 특화 에이전트 전망이 조직에 주는 의미

이 전망은 조직이 이미 쓰는 업무 애플리케이션에 에이전트 기능이 붙어서 들어온다는 뜻입니다. 그 결과 조직은 애플리케이션마다 따로 붙어 들어오는 에이전트와, 조직 고유의 업무 규칙을 담아 직접 만드는 에이전트를 함께 관리하게 됩니다. 두 번째 유형은 조직의 업무 담당자가 규칙을 정의해야 만들 수 있으므로, 업무 담당자가 에이전트 제작에 참여하는 방법이 함께 필요해집니다.

2.2.2 2028년 기업 소프트웨어 33%의 에이전틱 AI 포함과 일상 업무 결정 최소 15% 자율 처리 전망

기업용 소프트웨어의 에이전틱 AI 포함 비율 33%

Gartner는 2024년 10월 21일 발표한 「2025 전략 기술 트렌드: 에이전틱 AI」에서 2028년까지 기업용 소프트웨어 애플리케이션의 33%가 에이전틱 AI를 포함할 것으로 전망했습니다. 2024년에 이 비율은 1% 미만이었습니다[S5]. 2.2.1의 2026년 전망이 업무 특화 에이전트의 통합을 다뤘다면, 이 전망은 소프트웨어 자체가 여러 에이전트의 판단과 실행을 기본 기능으로 갖추는 방향을 다룹니다.

일상 업무 결정의 최소 15% 자율 처리

같은 날 Gartner는 2028년까지 일상 업무 결정의 최소 15%가 에이전틱 AI를 통해 자율적으로 처리될 것으로 전망했으며, 2024년의 비율은 0%였습니다[S6]. 이 전망에서 중요한 단어는 「결정」입니다. 에이전트가 답변을 만드는 데 그치지 않고 업무 처리 과정의 판단 일부를 맡는다는 뜻이기 때문입니다. 다만 결정을 맡기려면 판단 기준이 명확해야 하고 결과를 사람이 확인할 수 있어야 하므로, 이 전망은 업무 규칙을 어떻게 정의하고 점검할지에 대한 준비를 함께 요구합니다.

2028년 전망(Gartner, 2024-10-21)	2024년	2028년
에이전틱 AI를 포함한 기업용 소프트웨어 애플리케이션	1% 미만	33%
에이전틱 AI가 자율 처리하는 일상 업무 결정	0%	최소 15%

2.2.3 2030년 418억 달러 규모로 예상되는 기업용 에이전틱 AI 소프트웨어 시장

Omdia가 전망한 기업용 에이전틱 AI 소프트웨어 시장

Omdia는 2025년 9월 22일, 기업용 에이전틱 AI 소프트웨어 시장이 2025년 약 15억 달러에서 2030년 418억 달러로 커질 것으로 전망했습니다. 5년 사이 시장 규모가 약 28배로 커진다는 전망이며, Omdia는 이를 에이전틱 AI가 기존 생성형 AI보다 빠르게 성장한다는 분석으로 제시했습니다[S8]. 이 수치는 Omdia 보도자료를 근거로 한 값이므로, 다른 문서에 인용할 때는 원문 보고서와 대조하기를 권합니다.

시장 규모 전망을 읽는 방법

시장 규모 전망은 조사 기관마다 대상 범위와 산정 방식이 달라 수치를 서로 바로 비교하기 어렵습니다. 그러므로 이 전망에서 의사결정권자가 읽을 정보는 정확한 금액보다 기업용 에이전틱 AI 소프트웨어가 별도의 시장으로 집계될 만큼 커지고 있다는 방향입니다. 공급 측 투자가 이 방향으로 모이면 에이전트를 만들고 운영하는 도구도 빠르게 늘어나므로, 조직은 도구를 고르는 기준을 미리 정해 둘 필요가 있습니다.

2.3 PoC 뒤 중단 최소 30%·에이전틱 AI 프로젝트 취소 40% 이상이라는 반대편 신호

AI 투자에 대한 전망에는 확산 신호와 함께 실패 신호도 분명하게 들어 있습니다. Gartner는 2024년 7월 29일, 2025년 말까지 생성형 AI 프로젝트의 최소 30%가 개념 검증(PoC, Proof of Concept) 뒤 중단될 것으로 전망했고, 그 원인으로 데이터 품질, 위험 통제, 비용, 불분명한 가치를 꼽았습니다[S3]. 이어 2025년 6월 25일에는 2027년 말까지 에이전틱 AI 프로젝트의 40% 이상이 비용 상승, 불분명한 비즈니스 가치, 부족한 위험 통제 때문에 취소될 것으로 전망했습니다[S4]. 두 전망이 꼽은 원인이 거의 같다는 사실은 실패가 모델 성능보다 도입 방식에서 나온다는 점을 보여 줍니다.

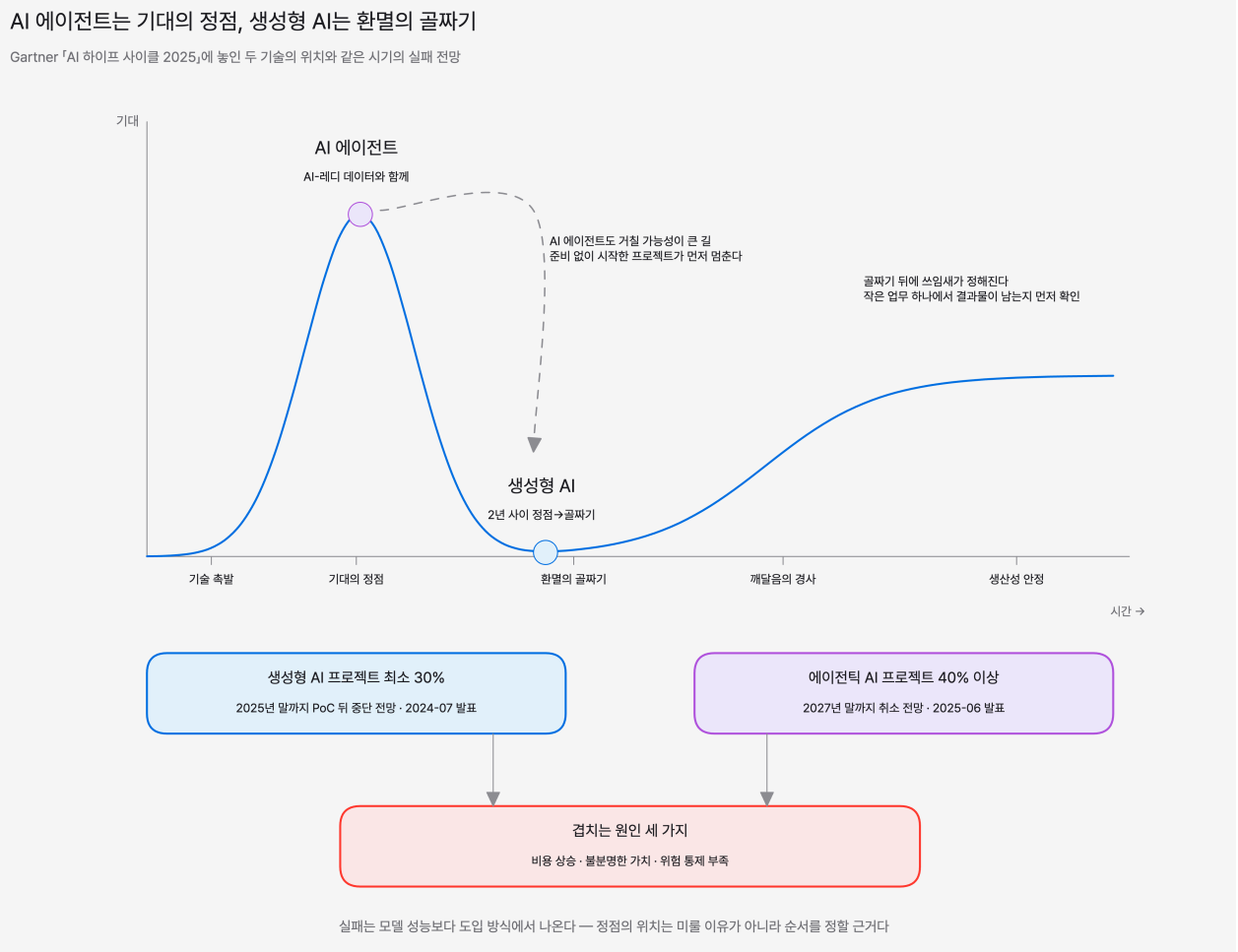


그림. Gartner 「AI 하이프 사이클 2025」에서 기대의 정점에 선 AI 에이전트와 환멸의 골짜기에 든 생성형 AI, 그리고 같은 원인을 꼽은 두 실패 전망(2024-07·2025-06)

2.3.1 기대의 정점에 선 AI 에이전트와 환멸의 골짜기에 든 생성형 AI

AI 하이프 사이클 2025의 두 위치

하이프 사이클(Hype Cycle)은 Gartner가 새 기술이 등장한 뒤 기대가 부풀었다가 꺼지고 실제 쓰임새가 정해지는 과정을 다섯 단계 곡선으로 나타낸 도구입니다. Gartner 「AI 하이프 사이클 2025」에서 생성형 AI는 환멸의 골짜기(Trough of Disillusionment)에 들어섰고, AI 에이전트는 AI-레디 데이터와 함께 기대의 정점(Peak of Inflated Expectations)에 놓였습니다[S9]. 생성형 AI가 골짜기로 내려간 이유를 Gartner는 조직들이 그 기술의 가능성과 한계를 함께 이해하게 되었기 때문이라고 설명했습니다[S9].

기대의 정점이 의사결정에 주는 경고

AI 에이전트가 기대의 정점에 있다는 것은 지금 쏟아지는 기대가 실제 성과보다 크다는 뜻입니다. 따라서 생성형 AI가 2년 사이 겪은 기대와 실망의 흐름을 AI 에이전트도 거칠 가능성이 높고, 그 과정에서 준비 없이 시작한 프로젝트가 먼저 중단됩니다. 의사결정권자는 이 위치를 도입을 미룰 이유로 읽기보다, 작은 업무 하나에서 결과물이 남는지를 먼저 확인하고 넓혀 가는 순서를 정할 근거로 읽는 편이 실용적입니다.

2.3.2 비용 상승·불분명한 가치·위험 통제 부족이 요구하는 도입 전 점검 항목

취소 원인 세 가지와 원인별 점검 질문

Gartner가 꼽은 에이전틱 AI 프로젝트 취소 원인 세 가지는 비용 상승, 불분명한 비즈니스 가치, 부족한 위험 통제입니다[S4]. 세 원인은 모두 프로젝트를 시작하기 전에 질문으로 바꿔 점검할 수 있으며, 시작 뒤에 발견하면 되돌리는 비용이 커집니다. 아래 표는 원인별로 도입 전에 답해 둘 질문을 정리한 것입니다.

취소 원인(Gartner, 2025-06-25)	도입 전 점검 질문
비용 상승	모델 호출 비용이 업무 처리 건수에 비례해 늘어날 때 상한을 어떻게 정하는가. 조직 안에서 운영하는 모델로 처리할 수 있는 업무인가
불분명한 비즈니스 가치	결과물이 업무 시스템에 남는 업무를 골랐는가. 처리 건수와 담당자 확인 시간처럼 셀 수 있는 지표가 있는가
부족한 위험 통제	에이전트가 쓰는 권한과 사람이 승인할 단계가 정해져 있는가. 처리 기록을 남기고 나중에 확인할 수 있는가

생성형 AI PoC 중단 원인과의 공통점

생성형 AI 프로젝트의 최소 30%가 PoC 뒤 중단될 것이라는 2024년 전망도 데이터 품질, 위험 통제, 비용, 불분명한 가치를 원인으로 꼽았습니다[S3]. 두 전망을 나란히 놓으면 데이터 품질을 제외한 세 원인이 겹칩니다. 그래서 위 표의 질문은 에이전틱 AI에 한정된 점검 항목이라기보다 AI 투자 전반에 적용되는 최소한의 사전 점검이며, 이 백서는 4장에서 비용을, 9장에서 위험 통제를, 10장에서 가치 측정을 다시 다룹니다.

2.4 답변 품질 경쟁에서 업무 완결 경쟁으로 — 유행과 구조 변화를 가르는 판단 근거

AI 경쟁의 기준은 어떤 모델이 더 좋은 답변을 내놓느냐에서 어떤 시스템이 업무를 끝까지 처리하느냐로 옮겨 가고 있습니다. 확산 전망과 실패 전망을 함께 놓으면, 결과를 가르는 요인은 모델의 답변 품질보다 업무에 정착했는지 여부라는 점이 드러납니다[S1][S7]. 이 기준은 조직이 AI 도입에서 부딪히는 세 가지 벽과 직접 연결되며, 3장에서 조직의 현재 단계를 판정할 때의 출발점이 됩니다.

2.4.1 조직 AI 도입의 세 가지 벽과 AI Agent가 겨냥하는 지점

조직 AI 도입을 막는 세 가지 벽

조직의 AI 도입이 시범 적용에서 멈추는 원인은 대개 모델보다 데이터 준비, 보안과 거버넌스, 인프라 비용이라는 세 가지 벽에 있습니다. 데이터 준비의 벽은 흩어지고 중복되고 권한이 뒤섞인 문서 위에서는 어떤 모델도 정확한 결과를 내기 어렵다는 문제입니다. 보안과 거버넌스(governance, AI가 무엇을 해도 되고 무엇을 하면 안 되는지 조직 차원에서 정하고 지키게 하는 관리 체계)의 벽은 AI가 민감 자료에 접근하는 순간 잘못된 답변보다 잘못된 노출이 더 큰 위험이 된다는 문제입니다. 인프라 비용의 벽은 사용량이 늘수록 모델 호출 비용이나 GPU 비용이 함께 늘어난다는 문제입니다.

세 가지 벽과 AI Agent의 대응

AI Agent는 세 가지 벽을 한 번에 없애지는 못하지만, 벽마다 넘는 방법을 업무 단위로 좁혀 줍니다. 아래 표는 각 벽에서 AI Agent가 겨냥하는 대응 방향을 정리한 것입니다.

도입의 벽	나타나는 모습	AI Agent의 대응 방향
데이터 준비	회사 전체 자료를 한꺼번에 정리하려다 시작하지 못한다	업무 하나가 실제로 참조하는 문서와 시스템만 먼저 연결한다
보안과 거버넌스	민감 자료가 외부로 나가거나 권한 밖 자료가 노출될 우려가 있다	조직 안에서 운영하고, 권한·승인 단계·처리 기록을 업무 단위로 정한다
인프라 비용	사용량에 비례해 모델 비용이 늘어난다	판단은 미리 만든 규칙과 코드에 맡기고, 모델은 요청을 해석하는 데 주로 써서 작은 모델로도 처리한다

세 대응 방향은 모두 업무 하나를 정해서 그 범위 안에서 데이터, 통제, 비용을 해결한다는 공통점이 있습니다. 이 가운데 보안과 비용의 벽은 4장에서 조직 안 sLLM(small Large Language Model, 조직 안 장비에서 운영할 수 있는 크기의 언어 모델)과 폐쇄망 운영 방식으로 다시 다룹니다.

시험 단계에 머무는 에이전트의 공통 조건

세 가지 벽을 넘은 뒤에도 에이전트가 시험 단계에 머무는 경우가 있습니다. 이런 에이전트에는 몇 가지 공통 조건이 보입니다. 첫째, 에이전트를 만든 사람과 그 결과를 쓰는 사람이 다르고, 결과를 쓰는 부서 담당자는 에이전트의 구성이나 실행 기록을 직접 볼 수 없습니다. 둘째, 에이전트가 무엇을 언제 실행했는지, 실패했을 때 누가 알게 되는지가 정해지지 않은 채 운영됩니다. 셋째, 시험이 끝난 뒤 업무 담당자가 그 에이전트를 넘겨받아 계속 쓰기 어려워, 에이전트가 만든 사람의 개인 도구로 남습니다. 이 세 조건은 모델의 성능과 관계없이 일을 시키는 창구와 지켜보는 창구가 없다는 데서 생기며, 7장에서 다루는 제품 정의의 출발점이기도 합니다.

2.4.2 업무 정착 여부로 판정하는 구조 변화의 기준

업무 정착을 판정하는 세 문항

AI Agent가 조직에 구조 변화를 만들었는지는 다음 세 문항으로 판정할 수 있습니다.

1. 결과물이 업무 시스템에 남는가 — 보고서·견적서·처리 기록처럼 다른 사람이 이어받아 쓰는 결과물이 저장되는가
2. 같은 업무가 반복 실행되는가 — 시범 사례 한 번으로 끝나지 않고 정해진 주기나 요청마다 계속 처리되는가
3. 업무 담당자가 규칙을 고치며 계속 쓰는가 — 업무 규칙이 바뀌었을 때 담당자가 직접 반영하고 에이전트가 바뀐 규칙으로 처리하는가

세 문항에 모두 「예」라고 답할 수 있는 업무가 하나라도 있으면, 그 조직에서 AI Agent는 일시적인 관심을 넘어 업무 구조의 일부가 된 것입니다.

판정 기준이 2장의 수치와 이어지는 방식

이 기준으로 2장의 수치를 다시 읽으면, 정기 사용 88%와 완전 확산 7% 사이의 간격은 세 문항을 통과한 업무가 아직 적다는 사실을 보여 줍니다[S1][S2]. 반면 2026–2030년 전망은 업무 특화 에이전트가 업무 애플리케이션과 일상 결정에 들어오는 방향을 가리키므로, 세 문항을 통과하는 업무는 앞으로 빠르게 늘어날 것입니다[S6][S7]. 결국 AI Agent가 유행인지 구조 변화인지는 산업 전체보다 각 조직 안에서 판정되며, 판정의 단위는

업무 하나입니다. 3장은 이 판정을 조직의 현재 AI 단계와 연결해, 지금 어느 단계에 있고 다음 단계로 가려면 무엇이 필요한지를 Prompt, Context, Loop, AI Agent, Agentic AI 다섯 단계로 비교합니다.

3장: Prompt·Context·Loop·AI Agent·Agentic AI 다섯 단계 비교

2장의 판정 단위가 업무 하나라면, 그 업무를 지금 AI에 어디까지 맡기고 있는지를 재는 잣대도 필요합니다. 조직의 AI 활용 수준은 Prompt, Context, Loop, AI Agent, Agentic AI 다섯 단계로 나눌 수 있고, 단계를 가르는 기준은 사람이 일의 어느 부분까지 직접 맡느냐입니다. 사내 챗봇을 이미 도입한 조직도 대부분 질문을 잘 쓰거나 사내 자료를 함께 넣어 답을 받는 앞의 두 단계에 머물러 있으며, 그 결과를 업무 시스템에 옮겨 적는 일은 여전히 사람이 합니다. 다음 투자를 정하려면 우리 조직이 지금 어느 단계에 있는지와 다음 단계로 가는 데 무엇이 모자란지를 먼저 확인해야 합니다. 다섯 단계 가운데 Loop는 계획·실행·확인·재시도를 AI가 스스로 반복하는 단계로 새로 정의한 것이고, 그 근거는 공개 논문과 모델 공급사의 설계 지침에서 가져왔습니다[S10][S11]. 마지막 절의 자가 진단표와 진입 조건표는 결재 단계에서 우리 조직의 AI 단계를 판단하는 첫 항목으로 그대로 쓸 수 있습니다.

3.1 사람의 개입 범위로 나눈 다섯 단계와 단계별 사람의 역할

다섯 단계는 AI에 붙는 기술이 하나씩 늘어나는 순서이면서, 동시에 사람이 손을 떼는 범위가 넓어지는 순서입니다. Prompt 단계의 사람은 매번 묻고 답을 옮겨 적고, Context 단계의 사람은 사내 자료가 붙은 답을 옮겨 적으며, Loop 단계부터는 목표를 주고 결과를 확인하는 역할로 바뀝니다. AI Agent 단계의 사람은 승인이 필요한 경우에만 승인하고, Agentic AI 단계의 사람은 여러 에이전트가 맡은 일의 목적과 결과를 관리합니다. 위 단계로 갈수록 AI가 혼자 처리하는 일이 늘어나므로 책임을 추적하는 장치와 사람이 개입할 승인 절차가 함께 필요해집니다.

단계	새로 붙는 기술	사람이 하는 일	견적 업무로 본 예
Prompt	질문을 잘 쓰는 방법	매번 묻고 결과를 옮겨 적는다	챗봇에 견적 안내 문구를 물어본다
Context	사내 자료를 함께 넣어 주는 검색	근거가 붙은 결과를 옮겨 적는다	사내 견적 규정을 찾아 답한다
Loop	계획·실행·확인·재시도 반복	목표를 주고 결과를 확인한다	금액을 계산하고 규칙에 맞는지 스스로 점검한다
AI Agent	도구·기억·권한	승인할 곳에서만 승인한다	CRM에 견적을 만들고 PDF를 저장한다
Agentic AI	여러 에이전트의 역할 분담과 조율	목적과 결과를 관리한다	영업기회 등록, 견적, 기술 문서가 이어서 처리된다



그림. Prompt에서 Agentic AI까지 다섯 계단과 계단마다 사람이 하는 일 — 매번 묻기·결과 입력·결과 확인·승인·목적 관리

3.1.1 질문 작성과 사내 자료 제공에 머무는 Prompt·Context 단계

Prompt 단계의 작업 방식

Prompt 단계는 LLM(Large Language Model, 대규모 언어 모델)에게 무엇을 원하는지 자연어로 잘 적어서 답을 받는 단계입니다. 웹 화면에서 생성형 AI 서비스에 질문을 입력하는 직원은 이미 이 단계에 있으며, 별도 개발이 필요 없어서 도입 장벽이 가장 낮습니다. 하지만 모델은 조직의 규정이나 고객 정보를 모르기 때문에 답의 정확성은 질문에 얼마나 많은 맥락을 직접 적어 넣었는지에 달려 있습니다. 받은 답을 업무 문서나 업무 시스템에 옮기는 일은 매번 사람이 하므로, 같은 일을 백 번 하면 사람의 손도 백 번 움직입니다.

Context 단계의 작업 방식

Context 단계는 질문과 관련된 사내 자료를 먼저 찾아 모델 입력에 함께 넣어 주는 단계이며, 흔히 RAG(Retrieval-Augmented Generation)라고 부르는 방식, 즉 질문과 관련된 문서를 검색해 답의 근거로 붙이는 방식이 여기에 속합니다. 모델을 다시 학습시키지 않고도 최신 규정·매뉴얼·계약 문서를 근거로 답하게 할 수 있어서, 사내 챗봇을 도입한 조직 대부분이 이 단계에서 시작합니다. 검색은 미리 정해 둔 고정 도구 하나이고 응답은 여전히 한 번 묻고 한 번 답하는 형태라서, 답을 바탕으로 업무 시스템을 고치거나 결과를 다시 점검하는 일은 일어나지 않습니다. 결과적으로 Context 단계는 답의 정확성을 끌어올리지만 사람이 결과를 옮겨 적는 일은 그대로 남깁니다.

구분	Prompt 단계	Context 단계
모델이 받는 입력	사람이 쓴 질문	질문과 검색해 붙인 사내 자료

구분	Prompt 단계	Context 단계
남는 한계	조직 사정을 모르고 결과는 사람이 옮긴다	답은 정확해지지만 결과는 여전히 사람이 옮긴다

3.1.2 목표를 주고 결과만 확인하는 Loop 단계로의 전환점

사람의 역할이 결과 입력에서 결과 확인으로 바뀌는 조건

Loop 단계로 넘어가면 사람은 한 번에 끝나는 질문을 쓰는 대신 이루어야 할 목표를 주고, AI는 그 목표를 이루기 위한 순서를 스스로 정해 실행합니다. 예를 들어 견적 금액을 계산하라는 목표를 받으면, AI는 품목 단가를 불러오고 계산한 뒤 정해 둔 규칙에 맞는지 점검하고, 맞지 않으면 계산을 고쳐 다시 점검합니다. 사람은 중간 계산을 옮겨 적지 않고 마지막에 나온 결과가 맞는지만 확인합니다. Context 단계와 Loop 단계를 가르는 조건은 모델의 크기와 무관하며, AI가 자기 결과를 확인하고 다시 시도하는 반복 구조를 갖췄는지로 정해집니다.

역할이 바뀐 뒤 사람이 새로 맡는 일

역할이 바뀌면 사람이 덜 일하게 되는 한편, 확인할 기준을 미리 정해 두는 일이 새로 생깁니다. AI가 스스로 점검하려면 무엇이 맞는 결과인지 판정할 규칙이 있어야 하고, 그 규칙은 업무를 아는 사람만 정할 수 있습니다. 그래서 Loop 단계부터는 업무 담당자가 「공공기관 견적은 부가가치세 포함으로 쓴다」처럼 판정 기준을 문장으로 정해 주는 일이 결과의 품질을 결정합니다.

구분	전환 전(Context 단계)	전환 후(Loop 단계)
사람이 주는 것	질문 한 줄	목표와 판정 기준
중간 결과 처리	사람이 옮겨 적고 고친다	AI가 확인하고 다시 시도한다
사람이 마지막에 하는 일	결과를 업무 문서로 옮긴다	결과가 맞는지 확인한다

3.2 계획·실행·확인·재시도를 스스로 반복하는 Loop 단계의 정의

Loop는 AI가 계획하고, 실행하고, 결과를 확인하고, 맞지 않으면 다시 하는 반복을 스스로 도는 단계입니다. 새로 세우는 용어이므로 정의를 받치는 근거를 세 가지로 밝혀 둡니다. 근거는 추론과 행동을 번갈아 하는 언어 모델 연구[S10], 에이전트를 도구 사용의 반복으로 설명하는 모델 공급사의 설계 지침[S11], 업계에서 하네스(Harness)라고 부르는 실행 골격입니다. 세 근거는 표현이 다르지만 모두 「매 단계 실제 결과를 받아 보고 다음 행동을 고친다」는 같은 구조를 가리킵니다.

근거	핵심 내용	Loop 정의와 맞닿는 부분
ReAct 논문(2022)	추론 기록과 행동을 번갈아 만들어 계획을 세우고 고친다[S10]	계획과 실행의 교차
공급사 설계 지침(2024)	에이전트는 환경의 피드백을 받으며 도구를 쓰는 반복이다[S11]	실행 결과에 따른 재시도
하네스 실행 골격	도구 호출·반복·검증을 엮는 실행 구조	확인과 재시도의 구현

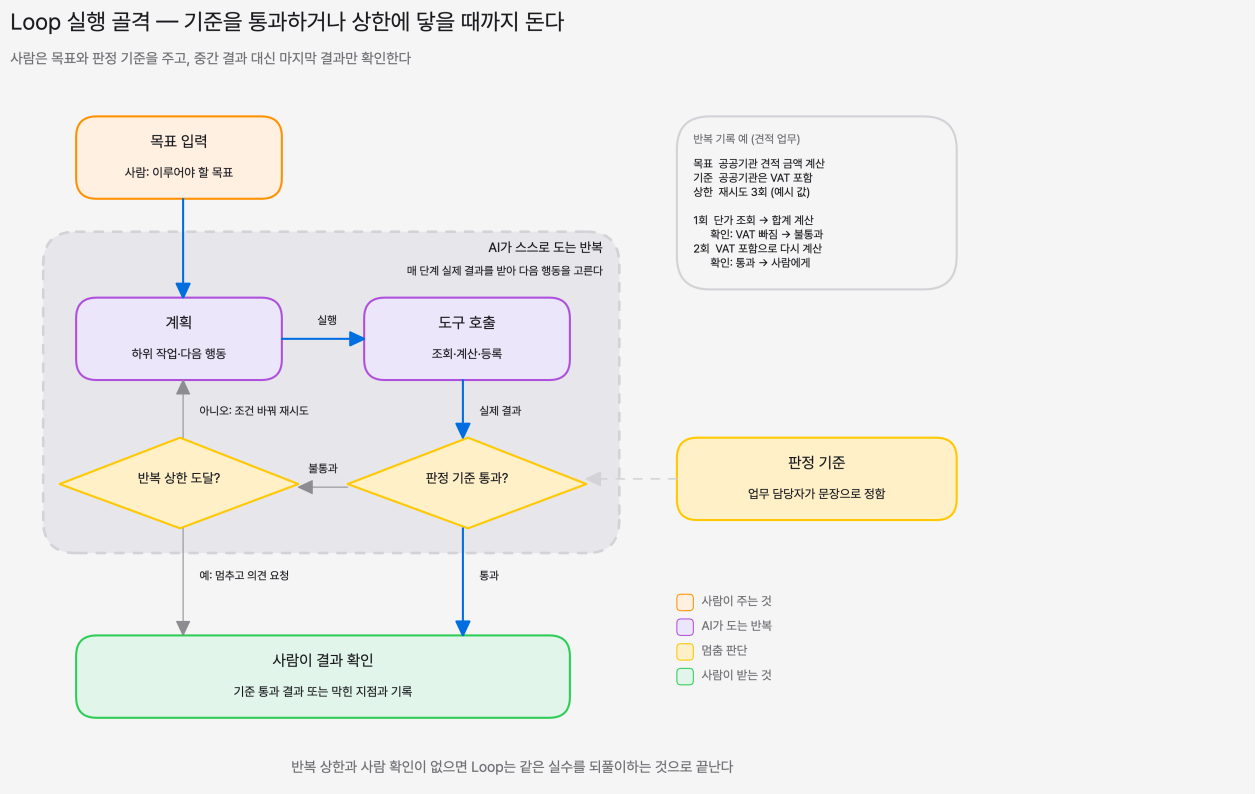


그림. 목표 입력 뒤 계획→도구 호출→결과 확인→재시도를 반복하고, 판정 기준을 통과하거나 반복 상한에 닿으면 사람에게 결과를 넘기는 Loop 실행 골격

3.2.1 ReAct의 추론·실행 교차와 매 단계 실제 결과를 받는 도구 사용 반복

ReAct 논문이 제시한 추론과 실행의 교차

Yao 등 7명의 연구진이 2022년 공개한 ReAct 논문은 언어 모델이 추론 기록과 업무 행동을 번갈아 생성하게 하는 방법을 제안했습니다[S10]. 이 방식에서 추론 기록은 모델이 행동 계획을 세우고, 진행 상황을 추적하고, 계획을 고치고, 예외를 처리하도록 돕습니다. 행동은 Wikipedia API 같은 외부 자료원에 실제로 질의해 새로운 정보를 가져오며, 그 결과가 다시 다음 추론의 입력이 됩니다. 즉 ReAct는 생각만 길게 하는 모델과 행동만 이어 가는 모델을 모두 넘어서, 계획과 실행을 한 고리 안에서 번갈아 처리하는 구조가 더 정확하다는 것을 보였습니다[S10].

설계 지침이 요구하는 매 단계의 실제 결과

Anthropic이 2024년 12월 공개한 설계 지침 「Building effective agents」는 미리 정해 둔 코드 경로로 모델과 도구를 엮은 시스템을 워크플로(workflow)로, 모델이 자기 처리 과정과 도구 사용을 동적으로 정하는 시스템을 에이전트로 나눕니다[S11]. 같은 지침은 에이전트를 환경의 피드백에 따라 도구를 쓰는 모델의 반복으로 설명하고, 실행 중에는 도구 호출 결과나 코드 실행 결과처럼 환경에서 나온 실제 결과를 매 단계 받아 진행 상황을 판단해야 한다고 강조합니다. 같은 지침은 반복 횟수 상한 같은 멈춤 조건과, 막히는 경우 사람에게 의견을 묻고 멈추는 확인 절차를 함께 두라고 권합니다[S11]. Loop 단계의 핵심은 모델의 답을 그대로 믿지 않고 실제 실행 결과로 확인한 뒤 다음 행동을 고르는 구조이며, 이 구조가 없으면 반복은 같은 실수를 되풀이하는 것으로 끝납니다.

3.2.2 6단계 분류의 Harness Engineering과 Loop 단계의 대응

여섯 단계 분류와 다섯 단계의 대응

AI Agent의 발전 과정을 Prompt Engineering, Context Engineering, Harness Engineering, AI Agent Framework, AI Agent Builder, AI Agent Orchestration의 여섯 단계로 나누는 분류도 함께 쓰이고 있습니다. 하네스는 모델에 도구 호출·반복·검증을 엮어 주는 실행 골격을 뜻하며, 앞에서 정의한 Loop와 같은 역할을 합니다. 다만 하네스라는 말은 구현 기술의 이름이라 의사결정권자에게는 낯설기 때문에, 이 백서는 사람이 체감하는 변화인 「반복」을 단계 이름으로 썼습니다.

다섯 단계	여섯 단계 분류	대응 방식
Prompt	1 Prompt Engineering	그대로 대응
Context	2 Context Engineering(RAG)	그대로 대응
Loop	3 Harness Engineering	실행 골격을 사람이 체감하는 반복으로 이름을 바꿈
AI Agent	4 AI Agent Framework, 5 AI Agent Builder	두 단계를 하나로 묶음
Agentic AI	6 AI Agent Orchestration	그대로 대응

Framework와 Builder를 AI Agent 하나로 묶은 이유

여섯 단계 분류의 Framework는 개발자가 코드로 에이전트를 조립하는 방식이고, Builder는 화면에서 블록을 이어 에이전트를 조립하는 방식입니다. 두 단계는 에이전트가 할 수 있는 일의 범위보다 누가 어떤 방법으로 에이전트를 만드는가를 나누는 기준에 가깝습니다. 그래서 다섯 단계 체계에서는 둘을 AI Agent 단계 하나로 묶고, 만드는 방법의 차이는 업무 담당자가 말로 정의하는 방식과 함께 5장·6장·8장에서 비교합니다.

3.3 한 업무를 끝까지 처리하는 AI Agent와 여러 에이전트를 조율하는 Agentic AI의 경계

AI Agent와 Agentic AI는 시장에서 자주 섞여 쓰이는 말이라서, 같은 제안서 안에서도 한 에이전트를 가리키다가 여러 에이전트의 협업을 가리키기도 합니다. 예를 들어 가트너는 2028년까지 기업용 소프트웨어 애플리케이션의 33%가 에이전틱 AI를 포함할 것으로 전망했는데(2024년 1% 미만), 이 전망 속 에이전틱 AI는 업무 특화 에이전트 하나를 넣은 경우까지 넓게 포함합니다[S5]. 이 백서는 두 말의 경계를 먼저 고정합니다. AI Agent는 도구·기억·권한으로 한 업무를 끝까지 처리하는 단위이고, Agentic AI는 여러 에이전트가 역할을 나눠 맡고 서로 조율하는 구성입니다.

구분	AI Agent	Agentic AI
처리 단위	업무 하나를 끝까지	여러 업무가 이어지는 흐름
갖추는 것	도구·기억·권한	역할 분담, 결과 전달, 조율
사람의 역할	승인할 곳에서만 승인	목적과 결과 관리
견적 업무로 본 예	견적 하나를 CRM에 만들고 PDF로 저장	영업기회 등록, 견적, 기술 문서 작성이 차례로 이어짐

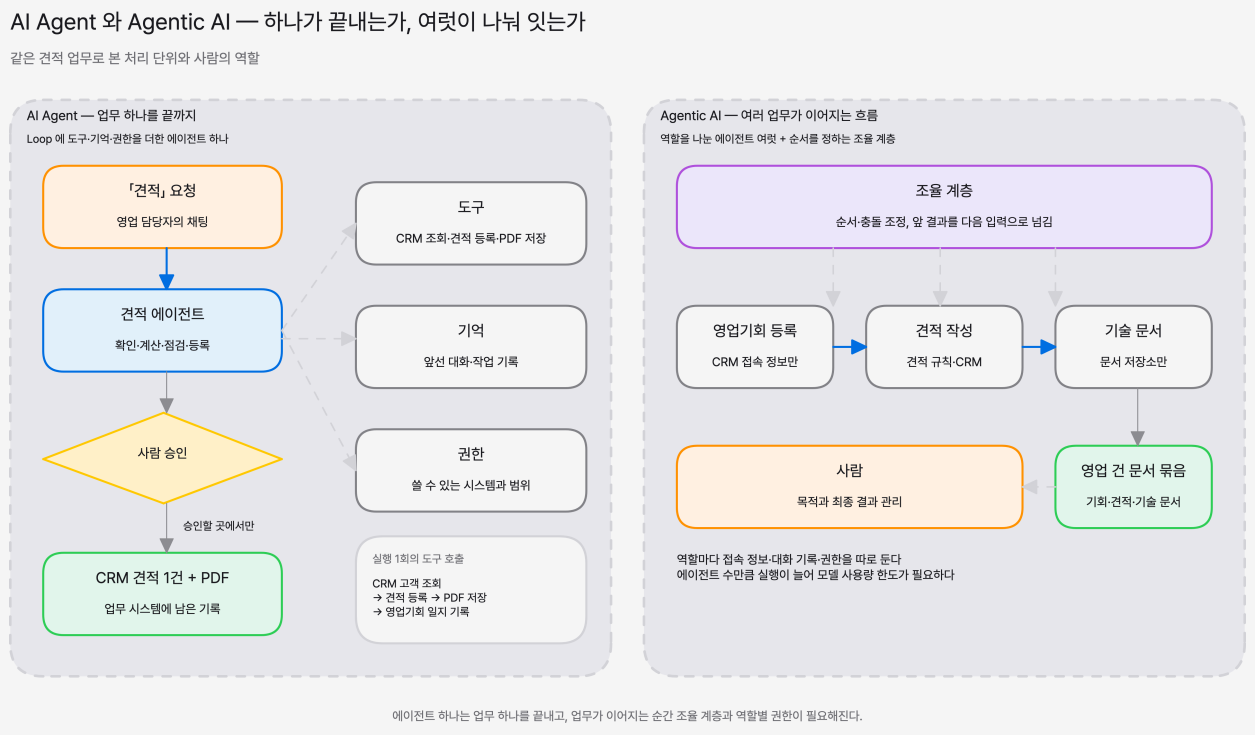


그림. AI Agent는 도구·기억·권한으로 업무 하나를 끝내고, Agentic AI는 역할을 나눈 에이전트들이 조율 계층 아래 결과를 주고받는다

3.3.1 도구·기억·권한으로 한 업무를 완결하는 AI Agent 단계

AI Agent가 갖추는 세 가지

AI Agent 단계는 Loop에 도구·기억·권한 세 가지가 더해져 업무 하나를 결과물까지 끝내는 단계입니다. 도구는 CRM, 문서 저장소, 메일처럼 업무 시스템을 실제로 읽고 쓰는 수단이고, 기억은 앞선 대화와 작업 기록을 이어받아 같은 설명을 반복하지 않게 하는 장치이며, 권한은 에이전트가 어느 시스템에 무엇을 쓸 수 있는지를 정한 범위입니다. 예를 들어 견적 업무를 맡은 AI Agent는 고객과 품목을 확인하고, 금액을 계산하고, 규칙을 점검한 뒤 CRM에 견적을 등록하고 PDF를 저장하는 데까지 한 번에 처리합니다. Loop 단계의 결과가 「확인할 계산」이라면 AI Agent 단계의 결과는 「업무 시스템에 남은 기록」입니다.

승인할 곳에서만 승인하는 사람의 역할

AI Agent 단계에서 사람은 모든 단계를 지켜보지 않고, 되돌리기 어려운 일이 일어나기 직전의 승인 단계에서만 판단합니다. 공급사 설계 지침도 에이전트가 정해 둔 확인 단계나 막힌 상황에서 멈추고 사람의 의견을 받도록 설계하라고 권합니다[S11]. OPENMARU AAS는 에이전트가 사람의 승인을 기다리는 요청을 승인 받은편지함 한곳에 모아 보여 주므로, 업무 담당자는 승인할 항목만 골라 처리할 수 있습니다[S12]. 따라서 AI Agent 단계로 가는 조직은 어느 업무 단계에서 사람이 반드시 승인할지를 에이전트를 만들기 전에 먼저 정해야 합니다.

한 업무를 나누어 처리하는 세 가지 방식

한 업무를 끝까지 처리하는 데 필요한 일이 많아지면 에이전트 플랫폼은 대개 세 가지 방식으로 일을 나눕니다. 첫째는 스킬로, 업무 절차를 문서 한 편과 보조 파일에 담아 두고 필요할 때 그 절차를 불러 쓰는 방식입니다. 둘째는 작업 보드로, 작업을 상태로 관리하면서 차단과 재개와 인계 기록을 남겨 작업자가 바뀌어도 앞선 결과를 이어받게 하는 방식입니다. 셋째는 하위 에이전트 위임으로, 실행 안에서 격리된 하위 실행을 띄우고 결과만 돌

려받는 방식입니다. 이 가운데 스킬은 AI Agent 단계 안에서 쓰이고, 작업 보드와 하위 에이전트 위임은 역할을 나눠 결과를 넘기는 다음 단계, 즉 Agentic AI 구성으로 넘어가는 지점에 있습니다.

3.3.2 역할 분담과 조율로 고정한 이 백서의 Agentic AI 정의

이 백서의 Agentic AI 정의

이 백서에서 Agentic AI는 여러 에이전트가 역할을 나눠 맡고, 한 에이전트의 결과를 다음 에이전트가 받아 이어서 처리하며, 그 순서와 충돌을 조율하는 구성을 가리킵니다. 예를 들어 영업기회를 등록하는 에이전트, 견적을 작성하는 에이전트, 기술 문서를 작성하는 에이전트가 같은 영업 건을 차례로 처리하고, 조율 계층이 각 결과를 다음 단계의 입력으로 넘깁니다. 사람은 개별 실행을 승인하는 일보다 전체 흐름의 목적이 맞는지와 최종 결과가 쓸 만한지를 관리하는 데 시간을 씁니다.

이 백서가 Agentic AI로 부르지 않는 뜻

시장에서는 Agentic AI를 AI가 얼마나 스스로 판단하는지를 나타내는 자율성의 정도로 쓰기도 하고, 업무 특화 에이전트 하나가 들어간 소프트웨어를 가리키는 말로도 씁니다[S5]. 하지만 같은 말이 단계 이름과 정도 표현으로 동시에 쓰이면 조직 안에서 「우리는 Agentic AI를 도입했다」는 보고가 무엇을 뜻하는지 확인할 수 없습니다. 이 백서는 자율성의 정도를 말할 때 「사람이 개입하는 범위」라는 표현을 쓰고, Agentic AI라는 이름은 여러 에이전트의 협업·조율에만 씁니다.

쓰는 뜻	쓰지 않는 뜻
여러 에이전트가 역할을 나눠 협업하고 조율하는 구성	AI가 스스로 판단하는 정도(자율성) 자체
결과를 다음 에이전트에 넘기는 업무 흐름	에이전트 기능 하나가 들어간 소프트웨어

역할을 나누는 이유 — 접속 정보와 대화 기록의 분리

여러 에이전트로 역할을 나누는 이유는 처리량보다 관리에 있습니다. 에이전트 하나에 영업 업무와 재무 업무를 함께 맡기면 두 업무에 필요한 접속 정보와 대화 기록이 한 에이전트 안에 쌓입니다. 이 경우 에이전트가 다른 업무의 맥락으로 답하거나, 맡은 일에 필요 없는 비밀 정보까지 갖게 됩니다. 조직이 사람의 일을 부서와 담당자로 나누는 것도 기록과 권한을 업무 단위로 갈라 두어야 누가 무엇을 보고 누가 결과를 책임지는지 정할 수 있기 때문입니다. Agentic AI 구성은 같은 원리를 에이전트에 적용한 것입니다. 역할을 나눌 때는 업무 범위와 함께 각 에이전트가 가질 접속 정보와 권한을 같은 단위로 정해 두어야 합니다.

여러 에이전트가 함께 일할 때 늘어나는 사용량

역할을 나누면 실행되는 에이전트의 수도 늘어납니다. 메신저 대화 한 건이나 예약 작업 한 건마다 에이전트가 따로 실행되고, 큰 조사를 맡은 에이전트는 하위 에이전트 여럿을 동시에 불러 일을 나눕니다. 여러 에이전트가 함께 일하는 구성은 에이전트 하나와 대화하는 방식보다 모델 사용량이 크게 늘어납니다. 나눠 맡길 큰 조사 업무가 있는 역할이라면, 그 역할이 쓸 모델의 종류와 사용량 한도를 IT 담당자가 먼저 정해 두는 편이 안전합니다.

3.4 업무가 요구하는 가장 낮은 단계에서 시작하는 단계 진단 원칙

단계를 고르는 원칙은 업무가 요구하는 가장 낮은 단계에서 시작한다는 것입니다. Anthropic의 설계 지침도 가능한 가장 단순한 해법을 먼저 찾고, 복잡성은 필요할 때만 더하라고 권합니다[S11]. 단계가 올라갈수록 비용과 응답 시간이 늘고 오류가 생겼을 때 원인을 찾기도 어려워지기 때문입니다. 가트너는 2027년 말까지 에이전틱

AI 프로젝트의 40% 이상이 비용 상승, 불분명한 비즈니스 가치, 부족한 위험 통제 때문에 취소될 것으로 전망했습니다[S4]. 예를 들어 단순 요약이라면 Prompt 단계로 충분하고, 사내 규정을 근거로 답해야 하면 Context 단계가 필요하며, 여러 시스템을 오가며 기록까지 남겨야 할 때 비로소 AI Agent 단계가 정당화됩니다.

3.4.1 진단 질문 다섯 개로 구성한 우리 조직 AI 단계 자가 진단표

자가 진단표를 읽는 방법

아래 다섯 질문은 단계 순서대로 놓여 있으며, 위에서부터 차례로 「예」라고 답할 수 있는 마지막 질문이 우리 조직의 현재 단계입니다. 업무마다 단계가 다를 수 있으므로 조직 전체를 한 번에 판정하기보다 대표 업무 하나를 정해 그 업무 기준으로 답하는 편이 정확합니다. 사내 챗봇이 규정을 찾아 답하지만 결과를 사람이 결재 문서로 옮기고 있다면, 2번까지 「예」이고 3번이 「아니오」이므로 그 업무는 Context 단계에 있습니다.

번호	진단 질문	「예」일 때의 단계
1	직원이 생성형 AI 서비스에 질문을 쓰고 받은 답을 업무에 쓰는가	Prompt
2	AI가 사내 규정·문서를 검색해 근거를 붙여 답하는가	Context
3	목표를 주면 AI가 계산과 점검을 스스로 반복해 확인된 결과를 내는가	Loop
4	AI가 업무 시스템에 권한을 갖고 기록·저장까지 끝내며 사람은 승인만 하는가	AI Agent
5	여러 에이전트가 업무를 나눠 맡고 결과를 서로 넘겨 이어서 처리하는가	Agentic AI

진단 결과를 결재 판단에 쓰는 방법

진단 결과는 다음 투자를 한 단계 위로 정하는 근거로 씁니다. 지금 Context 단계에 있는 업무에 곧바로 Agentic AI를 목표로 세우면, Loop와 AI Agent 단계에서 갖춰야 할 판정 기준과 권한 체계가 빠진 채 여러 에이전트가 연결됩니다. 따라서 결재 문서에는 현재 단계와 함께 바로 다음 단계와 그 단계에 필요한 조건을 같이 적는 편이 안전합니다.

3.4.2 Context 단계 조직의 다음 진입 조건 — 도구 연결·권한·결과 검증

Context 단계에서 Loop·AI Agent 단계로 가는 세 조건

사내 챗봇을 운영하는 조직은 대부분 Context 단계에 있으며, Loop와 AI Agent 단계로 가려면 도구 연결, 권한, 결과 검증 세 가지를 갖춰야 합니다. 도구 연결은 에이전트가 CRM이나 문서 저장소 같은 업무 시스템을 API로 읽고 쓸 수 있게 하는 일이며, 읽기 전용 연결부터 시작해 쓰기 연결로 넓히는 순서가 위험을 줄입니다. 권한은 에이전트가 쓸 수 있는 시스템과 범위, 그리고 사람이 승인해야 할 작업을 정하는 일입니다. 결과 검증은 업무 담당자가 정한 판정 기준을 코드로 옮겨 에이전트의 결과를 자동으로 점검하는 일이고, 매 단계 실제 결과로 진행을 확인하라는 설계 지침의 요구도 이 조건에 해당합니다[S11].

조건	확인할 내용	준비 주체
도구 연결	업무 시스템 API가 있는가, 읽기부터 열 수 있는가	IT 담당자
권한	에이전트 계정과 쓰기 범위, 사람 승인 단계를 정했는가	IT 담당자·업무 담당자
결과 검증	맞는 결과를 판정할 규칙을 문장으로 정했는가	업무 담당자

OPENMARU AAS가 세 조건을 지원하는 방식

OPENMARU AAS는 세 조건을 업무 담당자가 한 화면에서 준비하도록 돕습니다. 업무 시스템 연결은 MCP(Model Context Protocol, 모델이 외부 도구를 부르는 표준 규약) 서버와 스킬로 붙이고, 비밀값과 권한은 작업공간마다 따로 관리하며, 판정 기준은 업무 담당자가 말로 정한 규칙을 코딩 에이전트가 점검 코드와 시험으로 옮깁니다[S12]. 다음 단계 진입은 더 큰 모델을 고르는 일보다 업무 시스템 연결과 판정 기준을 갖추는 일에 달려 있으며, 이 준비가 끝난 업무 하나가 시범 도입의 첫 후보가 됩니다.

4장: 외부 LLM에 자료를 보내지 않는 내부 sLLM과 폐쇄망 AI Agent

3장에서 본 대로 AI Agent 단계로 가려면 업무 시스템과 도구를 연결해야 하고, 그 순간 모델에 실리는 자료의 범위가 달라집니다. sLLM(small Large Language Model, 조직 장비에서 직접 서빙할 수 있는 수십억–수백억 파라미터 규모의 언어 모델)은 성능만 보면 외부 LLM 서비스보다 뒤처지는 경우가 많습니다. 그런데도 AI Agent를 조직 안의 sLLM 위에서 실행해야 하는 이유는 AI Agent가 챗봇보다 훨씬 넓은 업무 자료를 모델에 보내기 때문입니다. 고객 정보, 견적 금액, 계약 문서가 도구 호출 결과로 모델 입력에 실리면, 외부 LLM을 쓰는 순간 그 자료가 매 호출마다 조직 밖으로 나갑니다. 외부 LLM과 내부 sLLM은 자료 반출·비용 구조·성능·규정 네 기준으로 비교할 수 있고, 작은 모델의 약점은 설계로 메우며, OPENMARU AAS는 모든 모델 호출이 사내 모델 게이트웨이를 거치는 폐쇄망 구성을 기준으로 삼습니다[S13]. 마지막 절의 예시 모델 장비 판정표는 장비 추가 구매가 필요한지를 가늠하는 출발점으로 쓸 수 있습니다.

4.1 질문 한 줄에서 업무 자료 전체로 넓어지는 AI Agent의 모델 전송 범위

챗봇이 모델에 보내는 자료는 사용자가 쓴 질문과 검색으로 붙인 문서 일부에 그칩니다. 반면 AI Agent는 목표를 이루는 동안 업무 시스템을 여러 번 조회하고, 조회 결과를 다음 판단의 입력으로 다시 모델에 보냅니다. 같은 업무라도 AI Agent가 모델에 보내는 자료의 양과 종류가 챗봇보다 크게 늘어나며, 자료 반출 문제는 사용 규정을 고쳐서 풀 수 있는 수준을 넘어 시스템 구조에서 생기는 문제가 됩니다.

구분	챗봇	AI Agent
모델이 받는 자료	질문, 검색으로 붙인 문서 일부	질문, 도구 호출 결과, 앞선 작업 기록
업무 자료의 예	사내 규정 문단	고객 정보, 견적 금액, 계약 문서, 업무 시스템 기록
한 업무당 모델 호출	대개 한 번	계획·실행·확인을 반복하는 만큼 여러 번

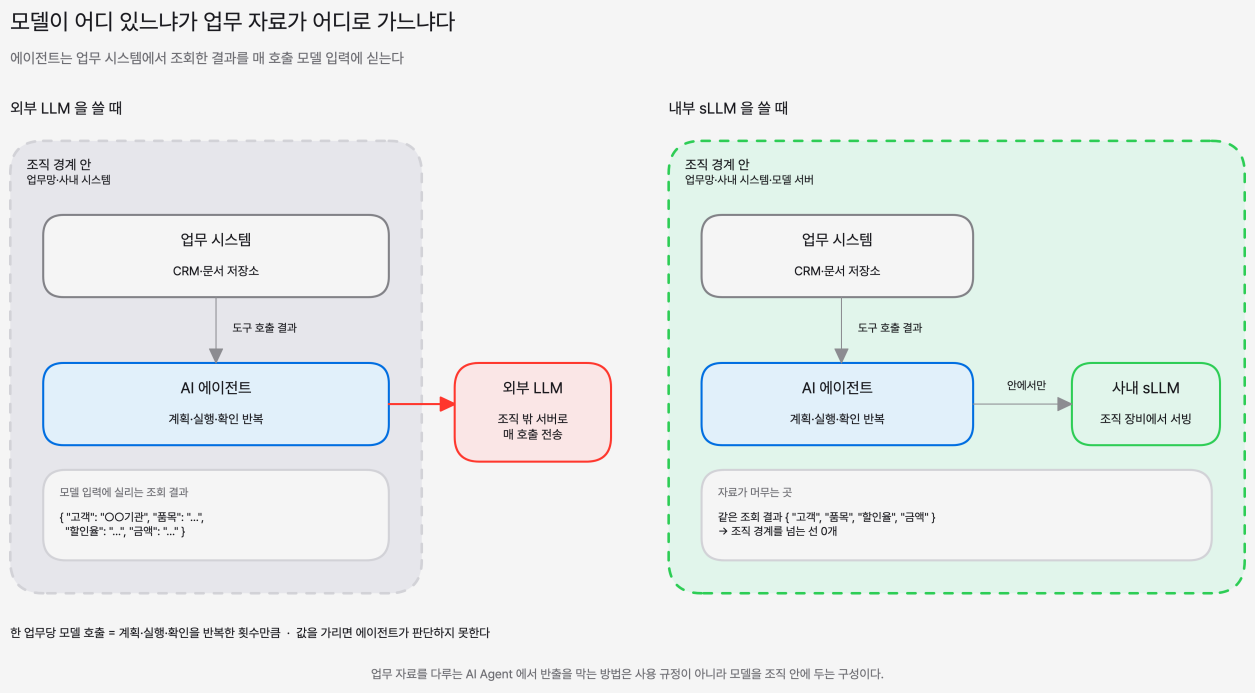


그림. 외부 LLM을 쓰면 도구 호출 결과가 매 호출 조직 밖으로 나가고, 내부 sLLM을 쓰면 같은 자료가 조직 안에 머문다

4.1.1 도구 호출 결과가 모델 입력에 그대로 실리는 에이전트 구조

도구 호출 결과가 다음 추론으로 들어가는 흐름

에이전트는 도구를 부를 때마다 그 결과를 관찰 내용으로 받아 다음 행동을 고르며, 이 관찰 내용은 다음 모델 호출의 입력에 그대로 들어갑니다[S10]. 모델 공급사의 설계 지침도 에이전트가 매 단계 도구 호출 결과 같은 실제 결과를 받아 진행 상황을 판단해야 한다고 설명합니다[S11]. 예를 들어 견적 에이전트가 CRM에서 고객의 지난 견적을 조회하면, 고객 이름과 품목, 할인율, 금액이 담긴 조회 결과가 그대로 모델 입력이 되고 모델은 그것을 읽고 새 견적의 단가를 정합니다. 이 과정이 외부 LLM 위에서 실행되면 고객 이름과 금액이 담긴 조회 결과가 호출마다 조직 밖 서버로 전송됩니다.

반출을 막는 방법이 구조에 있는 이유

도구 호출 결과에서 민감한 값만 골라 가리는 방법도 생각할 수 있지만, 에이전트는 그 값을 읽어야 판단할 수 있으므로 가리는 만큼 업무를 처리하지 못합니다. 견적 금액을 가린 채 견적 금액의 오류를 찾으라고 할 수는 없기 때문입니다. 그래서 업무 자료를 다루는 AI Agent에서 자료 반출을 막는 확실한 방법은 모델 자체를 조직 안에 두는 것이고, 이 판단은 개별 사용자의 주의나 사용 규정보다 시스템 구성 단계에서 내려야 합니다.

4.1.2 예산과 규정에 막히는 외부 LLM 서비스 사용

외부 LLM 서비스를 막는 세 가지 제약

외부 LLM 서비스가 아무리 뛰어나도 조직의 예산과 규정이 허락하지 않으면 업무에 쓸 수 없습니다. 이 제약은 성능 비교로 넘을 수 있는 문제와 성격이 다르며, 조직이 바꿀 수 없는 조건으로 먼저 확인해야 합니다. 대표적인 제약은 다음 세 가지입니다.

- 예산 제약 — 사용량에 따라 매달 금액이 달라지는 외부 서비스 요금은 연 단위로 고정된 예산 항목에 넣기 어렵습니다
- 보안 규정 제약 — 개인정보나 영업 비밀 등급이 높은 자료는 조직 밖으로 보내지 못하도록 정한 내부 규정이 있습니다
- 망 분리 제약 — 업무망이 인터넷과 분리된 기관은 외부 LLM 주소에 접속하는 것 자체가 불가능합니다

제약이 하나라도 있을 때의 판단

위 세 제약 가운데 하나라도 해당하는 업무는 비용이나 성능과 관계없이 내부 sLLM이 사실상 유일한 선택지입니다. 공공기관과 금융기관처럼 망 분리가 기본인 조직에서는 외부 LLM을 전제로 설계한 AI Agent 제품을 시범 도입하는 단계부터 진행할 수 없습니다. 따라서 의사결정권자는 성능 비교표보다 먼저 「이 업무의 자료가 조직 밖으로 나가도 되는가」를 확인하고, 답이 「아니오」라면 내부 sLLM을 전제로 제품을 고르는 편이 검토 시간을 줄입니다.

4.2 「sLLM은 성능이 낮다」는 반론과 판단은 코드·요청 해석은 모델이 맡는 설계

외부 LLM이 내부 sLLM보다 더 똑똑하다는 반론은 사실입니다. 최신 대형 모델은 긴 지시를 따르고 여러 도구를 연달아 고르는 능력에서 작은 모델보다 앞서며, 이 격차는 모델을 바꾸는 것만으로는 메워지지 않습니다. OPENMARU AAS가 택한 답은 판단을 미리 만든 코드에 맡기고, 모델은 사용자의 요청이 무엇을 뜻하는지 읽어 해당 코드를 실행하는 한 줄만 맡게 하는 설계입니다. 이렇게 역할을 나누면 작은 모델이 약한 긴 다단계 판단이 모델 밖으로 빠지므로, sLLM 위에서도 업무 결과가 매번 같게 나옵니다.

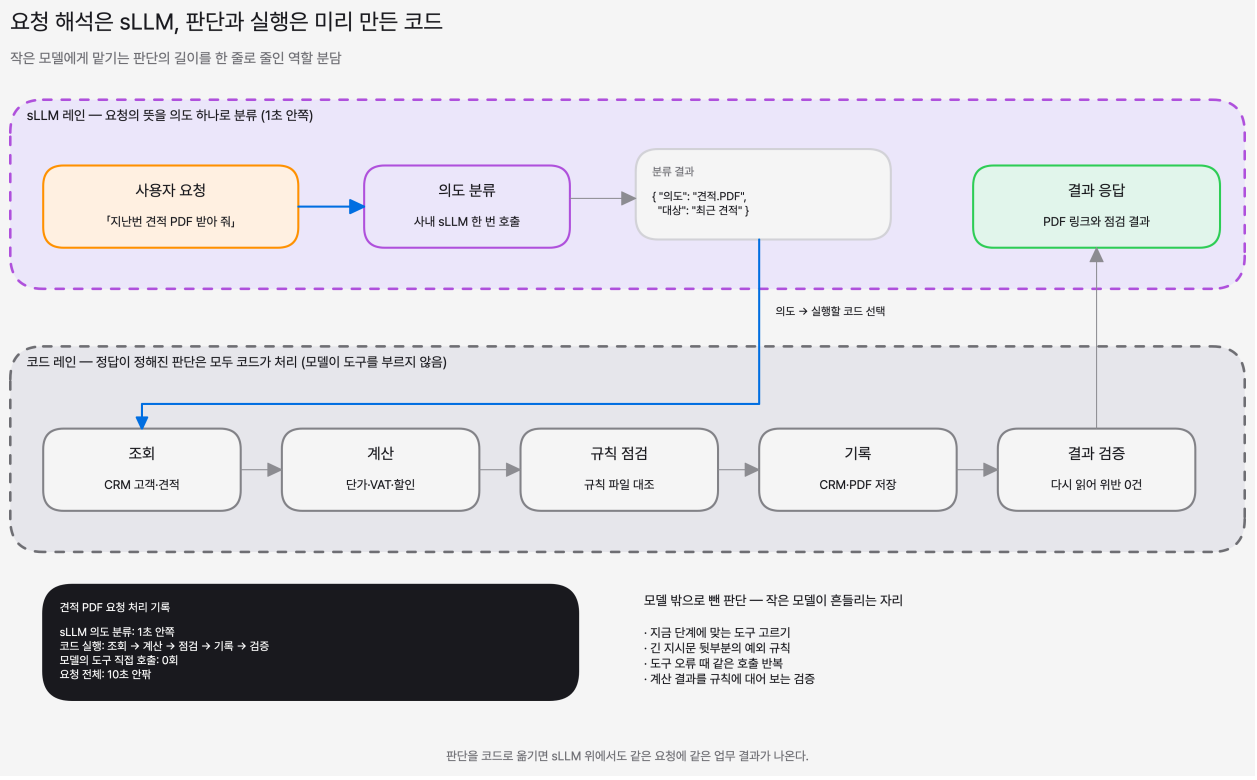


그림. 요청 해석은 sLLM이, 조회·계산·규칙 점검·업무 시스템 기록과 결과 검증은 미리 만든 코드가 맡는 역할 분담

4.2.1 긴 다단계 판단에서 흔들리는 작은 모델의 약점

작은 모델이 약한 판단의 유형

작은 모델의 약점은 단계가 길어질수록 커집니다. 에이전트가 도구 다섯 개를 차례로 골라야 하는 업무라면, 각 단계에서 생기는 작은 선택 오류가 다음 단계의 입력이 되어 쌓이고, 마지막 결과는 첫 단계의 작은 실수 하나로 전혀 다른 방향이 됩니다. 작은 모델에서 자주 나타나는 약점의 유형은 다음과 같습니다.

- 여러 도구 가운데 지금 단계에 맞는 도구를 고르는 판단이 매번 달라진다
- 긴 지시문의 뒷부분에 적힌 예외 규칙을 놓친다
- 도구가 오류를 돌려줬을 때 같은 호출을 반복하며 멈추지 않는다
- 계산 결과를 규칙에 대어 보는 검증 단계를 건너뛴다

약점을 인정한 뒤의 설계 방향

이 약점들은 모두 모델이 판단을 길게 이어 가야 할 때 생깁니다. 요청 하나를 읽고 그 뜻을 몇 가지 의도 가운데 하나로 고르는 일은 작은 모델도 짧은 시간 안에 안정적으로 처리합니다. 그래서 설계의 방향은 모델에게 맡기는 판단의 길이를 줄이는 것이며, 도구 선택·계산·규칙 점검처럼 정답이 정해진 판단은 코드로 옮기는 것입니다. OPENMARU AAS에서는 사내 모델이 1초 안쪽에 요청의 뜻을 가르고, 그 뒤의 조회와 계산, 결과 검증은 미리 만든 코드가 처리합니다. 예를 들어 견적 PDF를 받아 달라는 요청은 10초 안팎에 처리되며, 이 과정에서 모델이 도구를 직접 부르지 않습니다.

4.2.2 자료 반출·비용 구조·성능·규정으로 비교한 외부 LLM과 내부 sLLM 판단 기준표

네 기준에 따른 비교

외부 LLM과 내부 sLLM 가운데 어느 쪽이 나은지는 하나의 답으로 정해지지 않고, 자료 반출·비용 구조·성능·규정 네 기준에서 각각 유리한 쪽과 그 조건이 다릅니다. 아래 표는 네 기준별로 어느 쪽이 유리한지와 그 판단이 바뀌는 조건을 함께 적은 것입니다.

기준	외부 LLM	내부 sLLM	판단이 바뀌는 조건
자료 반출	매 호출마다 업무 자료가 조직 밖으로 나간다	자료가 조직 안에 머문다	다루는 자료가 공개 자료뿐이면 차이가 줄어든다
비용 구조	쓴 만큼 내고 초기 투자가 없다	장비와 운영 인력의 고정비를 낸다	물량과 가동률에 따라 유불리가 달라진다
성능	긴 다단계 판단에서 앞선다	요청 해석은 충분하고 긴 판단은 약하다	판단을 코드로 옮기면 격차가 업무 결과에 덜 드러난다
규정	망 분리·반출 금지 규정에 걸린다	망 분리 환경에서도 쓸 수 있다	규정이 외부 전송을 금지하면 다른 기준과 무관하게 내부 sLLM

표를 읽는 순서

네 기준은 같은 무게로 비교하지 않고, 규정과 자료 반출을 먼저 보고 비용 구조와 성능을 나중에 보는 순서로 읽는 편이 맞습니다. 규정이나 자료 등급 때문에 외부 전송이 막히는 업무는 나머지 두 기준을 볼 필요 없이 내부 sLLM으로 정해지기 때문입니다. 외부 전송이 허용되는 업무에서만 비용과 성능을 비교하며, 이때도 성능 격차는 판단을 코드로 옮기는 설계로 줄일 수 있다는 점을 함께 고려합니다.

4.3 모든 모델 호출이 사내 모델 게이트웨이를 거치는 폐쇄망 AI Agent 구성

OPENMARU AAS에서 AI 에이전트는 LiteLLM을 거쳐 AI 모델을 부르며, LiteLLM은 여러 모델 서버 앞에서 호출을 받아 넘겨주는 오픈소스 모델 게이트웨이입니다[S13]. 모든 모델 호출이 이 게이트웨이 한 곳을 지나므로, 관리자는 어떤 모델이 쓰이는지와 호출 기록을 한곳에서 관리할 수 있습니다. 대화에 쓸 수 있는 모델 목록에는 운영자가 정한 이름 규칙에 맞는 게이트웨이 모델만 나타나고, 질문 가드레일과 위험한 명령 차단 두 안전장치는 화면에서 끌 수 없게 되어 있습니다[S13]. IT 담당자는 아래 점검표로 기존 장비와 네트워크가 이 구성과 맞는지 바로 확인할 수 있습니다.

점검 항목	확인할 내용
모델 주소	사내 모델 서버가 OpenAI 호환 주소를 제공하는가
게이트웨이	모든 모델 호출을 사내 모델 게이트웨이로 모을 수 있는가
문맥 길이	모델이 도구 호출과 64,000 토큰 이상의 문맥을 지원하는가
외부 요청	실행 중 외부망 접속이 없는가, 이미지 저장소를 사내에 둘 수 있는가
코딩 모델 대체	만드는 단계의 코딩 에이전트를 사내 코딩 모델로 바꿀 계획이 있는가

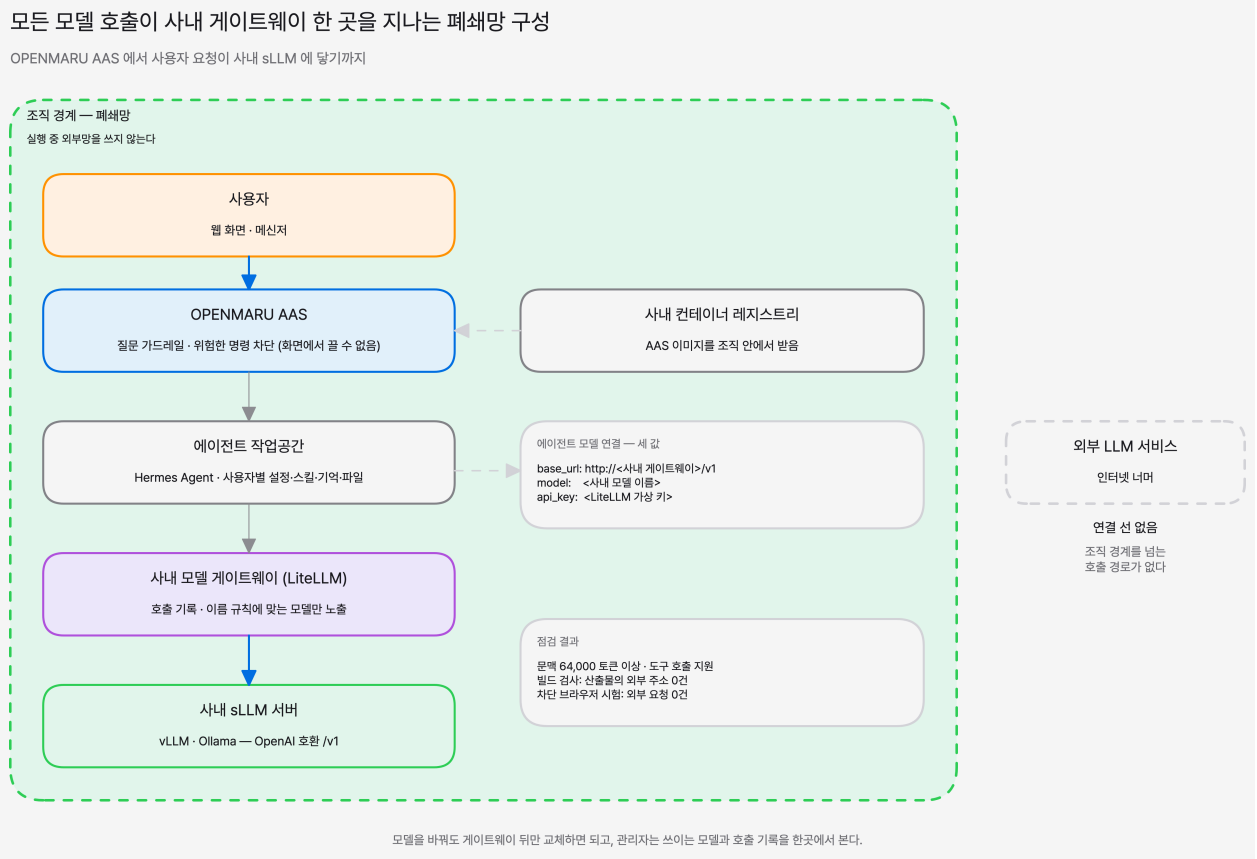


그림. 사용자 → OPENMARU AAS → 에이전트 작업공간 → 사내 모델 게이트웨이(LiteLLM) → 사내 sLLM 서버로 이어지는 폐쇄망 구성, 조직 경계 밖으로 나가는 선이 없다

4.3.1 OpenAI 호환 주소 하나로 연결하는 사내 모델 서빙

OpenAI 호환 주소 하나로 끝나는 연결

OPENMARU AAS와 그 아래에서 에이전트를 실행하는 오픈소스 Hermes Agent는 OpenAI 호환 주소 하나를 모델 연결로 씁니다[S14]. OpenAI 호환 주소는 OpenAI API와 같은 형식으로 요청을 받는 모델 서버 주소를 말하며, 사내에서 많이 쓰는 vLLM, LiteLLM, Ollama가 모두 이 형식을 지원합니다. 연결 설정은 서버의 `/v1` 주소, 모델 이름, 키 세 값을 설정 파일에 적는 것으로 끝나고, LiteLLM을 쓰면 키 자리에 LiteLLM이 발급한 가상 키를 적습니다[S14].

사내 모델 서버	연결 방식	설정할 때 확인할 것
vLLM	`/v1` 주소를 그대로 연결	도구 호출 옵션을 켜고 최대 문맥을 65,536 이상으로 실행
LiteLLM	게이트웨이 주소를 연결하고 가상 키 사용	뒤에 연결된 모델의 문맥 길이
Ollama	같은 장비나 내부망의 Ollama 주소 연결	문맥 길이를 65,536으로 실행하고 외부 접속 허용 주소 설정

기존 모델 서버를 그대로 쓰는 이점

연결 방식이 표준 주소 하나이므로, 이미 사내에 모델 서버를 운영하는 조직은 새 서빙 체계를 만들지 않고 기존 서버를 그대로 연결할 수 있습니다. 모델을 바꿀 때도 게이트웨이 뒤의 모델만 교체하면 되고 에이전트 쪽 설정은 바뀌지 않습니다. 따라서 IT 담당자가 먼저 확인할 것은 기존 서버가 OpenAI 호환 주소를 제공하는지 여부이며, 제공한다면 연결 작업 자체는 설정 몇 줄로 끝납니다.

4.3.2 도구 사용에 필요한 64,000 토큰 이상의 문맥 조건

문맥 길이가 도구 사용의 전제인 이유

Hermes Agent는 도구를 쓰려면 모델 문맥이 64,000 토큰 이상이어야 하며, 문맥이 그보다 작으면 에이전트가 일을 하지 않고 문맥이 작다는 안내만 돌려줍니다[S14]. 문맥은 모델이 한 번에 읽을 수 있는 입력의 최대 길이를 말하며, 에이전트는 지시문, 도구 설명, 앞선 대화, 도구 호출 결과를 모두 이 안에 담아야 합니다. 예를 들어 문맥 상한이 40,960 토큰인 모델은 다른 성능이 좋아도 이 조건을 채우지 못하므로 에이전트용으로 쓸 수 없습니다[S14].

모델을 고를 때 확인할 항목

내부 sLLM을 고르는 단계에서는 성능 순위보다 아래 세 항목을 먼저 확인하는 편이 시행착오를 줄입니다.

- 모델이 지원하는 최대 문맥이 64,000 토큰 이상인가
- 서빙 소프트웨어에서 그 문맥 길이로 실제로 실행되도록 설정했는가
- 모델과 서빙 소프트웨어가 도구 호출 형식을 지원하는가

세 항목 가운데 하나라도 빠지면 모델은 대화에는 답하지만 에이전트로서 일을 처리하지 못합니다. 특히 서빙 소프트웨어의 기본 문맥 길이가 모델의 최대값보다 작게 설정된 경우가 많으므로, 모델 사양서만 보고 판단하지 말고 실행 설정을 함께 확인해야 합니다.

4.3.3 외부 요청 0건을 빌드 때 검사하는 폐쇄망 배포

실행 중 외부망을 쓰지 않는 구성

OPENMARU AAS는 실행 중에 외부망을 쓰지 않습니다[S14]. 웹 화면에 필요한 글꼴과 3D 오피스 화면의 자산을 저장소 안에 함께 넣었고, 빌드 과정이 산출물에 외부 주소가 남아 있지 않은지 검사합니다. 네트워크를 막은 브라우저에서 3D 오피스 화면을 열어 외부 요청이 0건인지 확인하는 시험이 들어 있으며, 폐쇄망에서는 컨테이너 이미지를 조직의 자체 레지스트리에서 받도록 설정할 수 있습니다[S14].

점검 항목	확인 방법
화면 자산	글꼴과 3D 자산이 이미지에 포함돼 있는가
빌드 산출물	빌드 단계의 외부 주소 검사를 통과했는가
컨테이너 이미지	자체 레지스트리에 이미지를 올리고 그 주소를 설정했는가
모델 호출	모든 호출이 사내 모델 게이트웨이로 가는가

만드는 단계의 코딩 에이전트까지 조직 안에 두는 방법

OPENMARU AAS에서 업무 담당자가 말로 정한 규칙을 코드로 옮기는 만드는 단계에는 코딩 에이전트가 쓰이며, 현재는 외부 코딩 에이전트인 Claude Code를 연결해 쓰는 기능을 제공합니다[S12]. 이 단계는 규칙과 예시 자료만 다루고, 고객·금액 자료를 매일 처리하는 일하는 단계는 사내 sLLM 에이전트가 맡으므로 업무 자료는 조직 밖으로 나가지 않습니다. 다만 망 분리 기관처럼 만드는 단계의 외부 접속도 허용되지 않는 조직은 코딩 에이전트를 사내 코딩 모델로 바꿔 쓸 수 있으며, 이 경우 만드는 단계와 일하는 단계가 모두 조직 안에서 끝납니다.

4.4 예시 모델로 본 sLLM 장비 판정과 자체 서빙 비용의 판단 방향

내부 sLLM을 도입하려는 의사결정권자가 먼저 묻는 것은 지금 가진 장비로 충분한지, 새로 사야 하는지입니다. 이 질문의 1차 답은 모델 가중치 파일의 크기에서 나오며, 같은 모델이라도 양자화 수준에 따라 파일 크기가 세 배 넘게 달라집니다. 양자화(quantization)는 모델의 숫자 정밀도를 낮춰 파일 크기와 필요 메모리를 줄이는 방법을 말합니다. 아래 판정표는 공개된 27B 급 오픈웨이트 모델 하나를 예시 모델로 잡아 만든 것이며, 다른 모델을 고르면 수치는 달라집니다.

4.4.1 예시 모델의 양자화별 메모리와 추가 구매 여부

예시 모델 기준 장비 판정표

아래 표의 예시 모델은 Qwen 계열의 27B 급 오픈웨이트 모델이며, 파일 크기는 공개 모델 저장소에 게재된 값입니다. 모델 판본이 갱신되면 크기가 바뀔 수 있으므로 도입 검토 시점에 공개 모델 저장소에서 수치를 다시 확인해야 합니다.

양자화 수준(예시 모델)	가중치 파일 크기	장비 판정
4비트(Q4_K_M)	16.5GB	기존 장비 활용, 추가 구매 불필요
6비트(Q6_K)	22.0GB	추가 구매 불필요, 메모리 여유분 확인 필요
8비트(Q8_0)	29.0GB	가속기 1장 추가
BF16(원본 정밀도)	54.7GB	신규 도입

파일 크기만으로 승인하지 않는 이유

판정표의 장비 판정은 파일 크기만 놓고 본 1차 구분입니다. 모델을 서빙하려면 가중치 위에 문맥 캐시, 동시 처리 요청 수, 서빙 소프트웨어의 작업 메모리가 더해지며, 앞의 둘은 조직이 정하는 설정값입니다. 앞에서 본 64,000 토큰 이상의 문맥 조건[S14]을 채우고 여러 사용자가 동시에 요청하면 캐시가 그만큼 커지므로, 16.5GB 파일이라도 가속기 메모리가 넉넉해야 합니다. 따라서 결재 단계에서는 파일 크기와 함께 조직이 정한 문맥 길이와 동시 요청 수로 계산한 여유분을 요구하는 편이 안전합니다.

4.4.2 사용량과 가동률에 따라 달라지는 자체 서빙의 유불리

자체 서빙 비용이 시간에 따라 쌓이는 구조

외부 LLM은 호출이 없으면 요금도 없지만, 자체 서빙은 요청이 없는 동안에도 장비의 감가상각, 전력, 관리 비용이 시간 단위로 발생합니다. 자체 서빙의 토큰당 원가는 일정 기간의 고정비를 그 기간에 실제로 처리한 토큰 수

로 나눈 값이 되며, 처리한 물량이 적을수록 원가가 올라갑니다. 정리하면 물량이 일정 수준을 넘어야 자체 서빙이 유리하고, 가동률이 낮으면 자체 서빙이 오히려 비쌉니다. 유불리가 갈리는 정확한 물량은 비교 대상 외부 서비스의 단가, 장비의 가동률, 운영 인력 비용에 따라 크게 달라지므로 일반적인 수치로 정할 수 없고, 조직의 실제 호출량으로 판단해야 합니다.

구분	가동률 높음	가동률 낮음
물량 많음	자체 서빙이 유리하다	업무 시간대 집중 여부를 보고 장비 규모를 조정한다
물량 적음	장비를 작게 잡으면 자체 서빙도 검토할 수 있다	비용만 보면 외부 서비스가 유리하다

에이전트 물량을 대화 건수로 추정하지 않는 이유

표의 물량을 추정할 때는 에이전트의 사용량이 대화형 AI와 다르게 늘어난다는 점을 함께 봐야 합니다. 챗봇은 질문 하나에 모델 호출 한두 번으로 끝나지만, AI Agent는 계획·도구 호출·결과 확인을 되풀이하므로 요청 한 건에도 모델이 읽고 쓰는 글의 양이 여러 배가 됩니다. 도구가 돌려준 결과가 다음 호출의 입력에 그대로 실리고, 예약 작업은 사람이 화면을 보지 않는 시각에도 정해진 주기로 실행됩니다. 물량은 직원 수나 대화 건수보다 맡길 업무의 건수, 업무 한 건의 평균 도구 호출 횟수, 예약 작업의 실행 주기를 곱해 추정하는 편이 실제에 가깝습니다. 시범 도입 기간에 이 세 값을 실제로 재 두면 장비 규모를 다시 잡을 때 근거로 쓸 수 있습니다.

비용보다 앞서는 판단 기준

위 표는 비용만 놓고 본 판단이며, 4.1절의 규정이나 자료 등급 때문에 외부 전송이 막힌 업무에는 적용하지 않습니다. 그런 업무는 물량이 적어도 내부 sLLM으로 처리해야 하므로, 판단 근거를 비용에서 규정과 자료 통제로 바꿔 장비 규모를 정합니다. 가동률을 판단하려면 모델 호출량을 집계해야 하는데, 모든 호출이 사내 모델 게이트웨이를 거치는 구성에서는 게이트웨이의 호출 기록으로 이 값을 바로 확인할 수 있습니다[S13].

5장: AI Agent를 만드는 주체인 업무 담당자

4장까지가 AI Agent를 어디에서 실행할지에 대한 답이었다면, 도입하려는 조직이 다음으로 부딪히는 질문은 누가 그 에이전트를 만들 것인가입니다. 흔히 떠올리는 답은 AI 개발자를 새로 뽑거나 외부 업체에 구축을 맡기는 방식인데, 두 방식 모두 업무 규칙을 아는 사람과 코드를 짜는 사람이 달라서 규칙이 전달되는 과정에서 빠지고 바뀝니다. 2장에서 본 Gartner 전망은 에이전틱 AI 프로젝트의 40% 이상이 2027년 말까지 취소될 원인으로 비용 상승, 불분명한 업무 가치, 위험 통제 부족을 꼽았습니다[S4]. 업무 가치가 불분명해지는 원인의 상당 부분은 무엇이 맞는 결과인지를 정할 사람이 제작 과정 밖에 있다는 데서 생깁니다. OPENMARU AAS는 업무 담당자가 목적과 판정 기준을 말로 정하고, 코드와 시험은 코딩 에이전트가, 모델과 시스템 연동은 IT 담당자가 맡는 분담을 전제로 설계했습니다. 세 가지 제작 주체를 나란히 놓으면 이 분담이 왜 필요한지 드러나고, 업무 담당자가 말로 정하는 범위와 그렇지 않은 범위도 경계까지 나눌 수 있습니다.

5.1 AI 개발자 채용·외부 구축·업무 담당자 — 제작 주체 세 가지 비교

AI Agent의 제작 주체는 AI 개발자 채용, 외부 구축, 업무 담당자 직접 제작의 세 가지로 나눌 수 있습니다. 세 방식은 업무를 얼마나 정확히 이해하는지, 규칙이 바뀔 때 얼마나 빨리 고칠 수 있는지, 만든 뒤 유지하는 비용이 누구에게 쌓이는지에서 차이가 납니다. 에이전트는 한 번 만들고 끝나는 프로그램과 달리 업무 규칙이 바뀔 때마다 함께 고쳐야 하므로, 처음 제작 비용보다 변경 비용이 총비용을 더 크게 좌우합니다. 채용 계획을 세우기 전에 세 방식의 변경 경로를 먼저 비교해 보는 편이 판단에 도움이 됩니다.

5.1.1 업무 규칙을 개발자에게 다시 설명하는 전달 비용

업무 규칙의 전달 경로와 손실

규칙을 아는 사람과 코드를 짜는 사람이 다르면 규칙은 요구사항 문서, 구두 설명, 개발자의 해석을 차례로 거쳐 코드에 들어갑니다. 이 과정에서 「공공기관 견적은 VAT 포함」 같은 명시 규칙은 비교적 잘 전달되지만, 문서에 적힌 적이 없는 관행은 개발자가 물어볼 이유조차 모르기 때문에 빠지기 쉽습니다. VAT 포함 여부를 견적 제목 말머리와 출력 서식으로 표시하고 세액 칸에는 따로 적지 않는 관행이 그런 예로, 담당자에게는 당연하지만 개발자에게는 처음 듣는 규칙입니다. 결국 빠진 규칙은 운영 중 틀린 결과로 드러나고, 담당자가 다시 설명하고 개발자가 다시 고치는 왕복이 생깁니다.

전달 단계별로 생기는 손실 목록

전달 경로의 단계마다 생기는 손실을 정리하면 다음과 같습니다. 각 단계의 손실은 서로 쌓이므로, 단계 수가 줄어들수록 규칙이 원래 뜻대로 코드에 들어갈 가능성이 커집니다.

- 요구사항 작성 단계: 담당자가 당연하게 여기는 관행과 예외가 문서에서 빠집니다
- 설명과 해석 단계: 업무 용어의 뜻이 개발자의 일반 상식으로 바뀌어 해석됩니다
- 구현 단계: 규칙끼리 충돌할 때 어느 쪽을 앞세울지 개발자가 임의로 정합니다
- 수정 요청 단계: 틀린 결과를 담당자가 다시 설명하고 개발 대기열에서 순서를 기다립니다

전달 비용은 규칙이 바뀔 때마다 다시 치러야 하는 반복 비용입니다. 에이전트 공급사도 공식 글에서 가장 단순한 해법부터 시작하고 필요할 때만 복잡도를 올리라고 권하는데[S11], 전달 단계를 줄이는 일 역시 같은 방향의 단순화입니다.

5.1.2 업무 이해·변경 속도·유지 비용으로 본 제작 주체 비교표

세 제작 주체의 판정 결과

세 방식을 업무 이해, 변경 속도, 유지 비용의 세 기준으로 판정하면 업무 담당자가 직접 정하는 방식이 업무 이해와 변경 속도에서 앞섭니다. 다만 업무 담당자 혼자 모든 것을 맡는 구조는 권한 관리와 운영 기준이 부서마다 달라질 위험이 있으므로, 표의 셋째 열은 업무 담당자가 규칙을 정하고 IT 담당자가 기반을 관리하는 분담을 전제로 합니다.

기준	AI 개발자 채용	외부 구축	업무 담당자가 정하고 IT가 기반 제공
업무 이해	요구사항 문서로 전달받아 관행·예외가 빠지기 쉬움	계약 범위 안의 요구사항만 반영	규칙을 아는 사람이 직접 문장으로 정함
변경 속도	사내 개발 대기열 순서에 묶임	변경 요청마다 계약·견적 절차	규칙 문장을 고치면 코딩 에이전트가 코드와 시험을 함께 고침
유지 비용	개발자 인건비가 상시 발생, 퇴사 시 지식 유실	유지보수 계약에 종속	규칙이 파일과 사람이 읽는 페이지로 남아 담당자 교체 뒤에도 이어짐
보안·운영 기준	사내 기준 적용 쉬움	업체 기준과 사내 기준 조율 필요	IT 담당자가 모델·권한·연동 기준을 관리

표의 판정은 업무 담당자 방식이 다른 두 방식을 대체한다는 뜻이 아닙니다. 모델 서빙, 권한, 사내 시스템 연동은 여전히 IT 담당자의 일이고, 복잡한 시스템 통합에는 개발 인력이 필요합니다. 반면 업무 규칙을 정하고 결과가 맞는지 판정하는 일은 업무 담당자만 정확히 할 수 있으므로, 이 몫을 개발자에게 넘기지 않는 것이 변경 비용을 낮추는 핵심입니다. Gartner가 꼽은 에이전틱 AI 프로젝트 취소 원인 가운데 불분명한 업무 가치와 위험 통제 부족은[S4] 판정 기준을 정할 사람이 빠진 구조와 직접 이어집니다.

5.2 목적·대상·판정 기준·검수로 업무를 가르치는 에이전트 4계층 책임 구조

AI Agent는 모델, 도구, 맥락, 업무 규칙의 네 계층으로 나눠 볼 수 있습니다. 모델과 도구는 공급사와 IT 부서가 고르고 관리하는 계층이고, 맥락과 업무 규칙은 그 업무를 해 온 사람만 정확히 쓸 수 있는 계층입니다. 도입 후 「누가 무엇을 책임지는가」를 두고 혼선이 생기는 원인은 대부분 이 경계를 정하지 않은 채 시작하는 데 있습니다. 4계층마다 책임 주체를 미리 적어 두면 문제가 생겼을 때 어느 쪽이 고칠지도 함께 정해집니다.

계층	담는 내용	책임 주체
모델	기반 모델의 선택과 버전, 서빙 장비	IT 담당자(공급사 협력)
도구	사내 시스템 연동, 검색, 파일 처리 기능	IT 담당자
맥락	업무 목적, 처리 대상, 참조 자료, 예시	업무 담당자
업무 규칙	판정 기준, 예외 처리, 승인 조건	업무 담당자

5.2.1 업무 담당자의 몫인 목적 설정·판정 기준·검수

업무 담당자가 정하는 네 가지 항목

업무 담당자가 에이전트에게 업무를 가르칠 때 정하는 것은 목적, 대상, 판정 기준, 검수의 네 가지입니다. 이 네 가지는 코딩 문법과 무관하고, 그 업무를 몇 년 해 온 사람이라면 이미 머릿속에 갖고 있는 내용입니다. 견적 작성 업무를 예로 들면 네 항목은 다음처럼 채워집니다.

- 목적: 영업 담당자의 요청을 받아 사내 견적 규칙에 맞는 견적서를 CRM에 만들고 PDF로 남긴다
- 대상: 「견적」으로 시작하는 요청, 그리고 이미 등록된 영업기회에 딸린 견적
- 판정 기준: 공공기관 견적은 VAT 포함, 조달 견적은 할인 없이 등록가 그대로, 플랫폼 제품 판매에는 커스터마이징 비용 포함 여부 확인
- 검수: 만들어진 견적을 다시 읽어 규칙 위반이 없는지 확인하고, 틀린 결과가 나오면 무엇이 틀렸는지 말로 알려 준다

판정 기준 문장이 정하는 에이전트 품질

네 항목 가운데 에이전트 품질을 가장 크게 좌우하는 것은 판정 기준입니다. 대상을 넓게 잡으면 에이전트가 엉뚱한 자료까지 처리하고, 판정 기준이 흐리면 틀린 결과를 합격으로 내보냅니다. 「공공기관 견적은 VAT 포함」이라는 문장 하나가 있으면 코딩 에이전트는 이를 작성 전 점검 규칙으로 옮기고, 공공기관인데 VAT 별도로 작성된 견적을 만들기 전에 멈추게 합니다. 따라서 업무 담당자에게 필요한 훈련은 코딩 교육보다 목적 문장 쓰기, 대상 범위 정하기, 판정 기준 쓰기, 결과 검수하기의 네 단계 실습입니다.

5.2.2 IT 담당자의 몫인 모델·권한·연동·운영 환경

IT 담당자가 관리하는 네 가지 기반

IT 담당자는 업무 담당자가 정한 규칙이 안전하게 실행될 기반을 관리합니다. OPENMARU AAS 관리자 매뉴얼은 관리자, 팀 관리자, 작업공간 주인, 일반 사용자의 네 가지 권한을 두고, 모든 AI 에이전트가 LiteLLM 모델 게이트웨이를 거쳐 AI 모델을 부르는 구성을 기준으로 설명합니다[S13]. 어떤 모델을 쓰고, 누가 어떤 에이전트와 스킬에 접근하며, 어느 사내 시스템과 연결할지는 IT 담당자가 정하는 영역입니다. IT 담당자의 책임을 항목으로 나누면 다음과 같습니다.

- 모델: 사내 sLLM 서빙과 모델 게이트웨이 구성, 대화용 모델 목록의 이름 규칙 관리
- 권한: 사용자·부서 등록, 네 가지 권한 배정, 작업공간 한도 설정
- 연동: CRM·문서 저장소·메신저 같은 사내 시스템 연결과 비밀값 관리
- 운영 환경: 설치, 업그레이드, 사용량과 변경 기록 점검, 안전장치 유지

에이전트 플랫폼을 고치지 않는 연동 방식

OPENMARU AAS는 에이전트 플랫폼인 Hermes Agent를 직접 고치지 않습니다. 설정을 바꾸거나 대화할 때 Hermes 대시보드 API를 부르고, Hermes가 변경을 검사해 저장하며, AAS는 Hermes 데이터 폴더에 직접 쓰지 않습니다[S14]. 이 구조 덕분에 IT 담당자는 플랫폼 버전과 설정 기준을 한곳에서 관리할 수 있고, 업무 담당자가 스킬을 바꾸더라도 플랫폼이 검사하지 않은 변경이 저장되지 않습니다. 관리자 매뉴얼의 안전장치인 질문 가드레일과 위험한 명령 차단은 화면에서 끌 수 없게 되어 있어[S13], 업무 담당자가 만든 에이전트도 같은 통제 아래에서 실행됩니다.

5.3 「업무 담당자가 정말 만들 수 있나」라는 의문과 말로 정하는 범위의 경계

「업무 담당자가 정말 에이전트를 만들 수 있나」라는 의문은 정당합니다. 파이썬이나 node.js를 모르는 사람이 CRM 연동 코드와 점검 로직을 작성할 수는 없기 때문입니다. OPENMARU AAS가 전제하는 분담은 업무 담당자가 모든 것을 만든다는 그림과 다릅니다. 업무 담당자는 규칙을 말로 정하고, 코드와 시험은 코딩 에이전트가 만들며, 시스템 연동은 IT 담당자가 맡습니다. 다만 스크립트가 필요한 스킬은 지금 AAS 화면에서 직접 고칠 수 없고 코딩 에이전트 연결을 거쳐야 한다는 점은 함께 알아 둘 조건입니다.

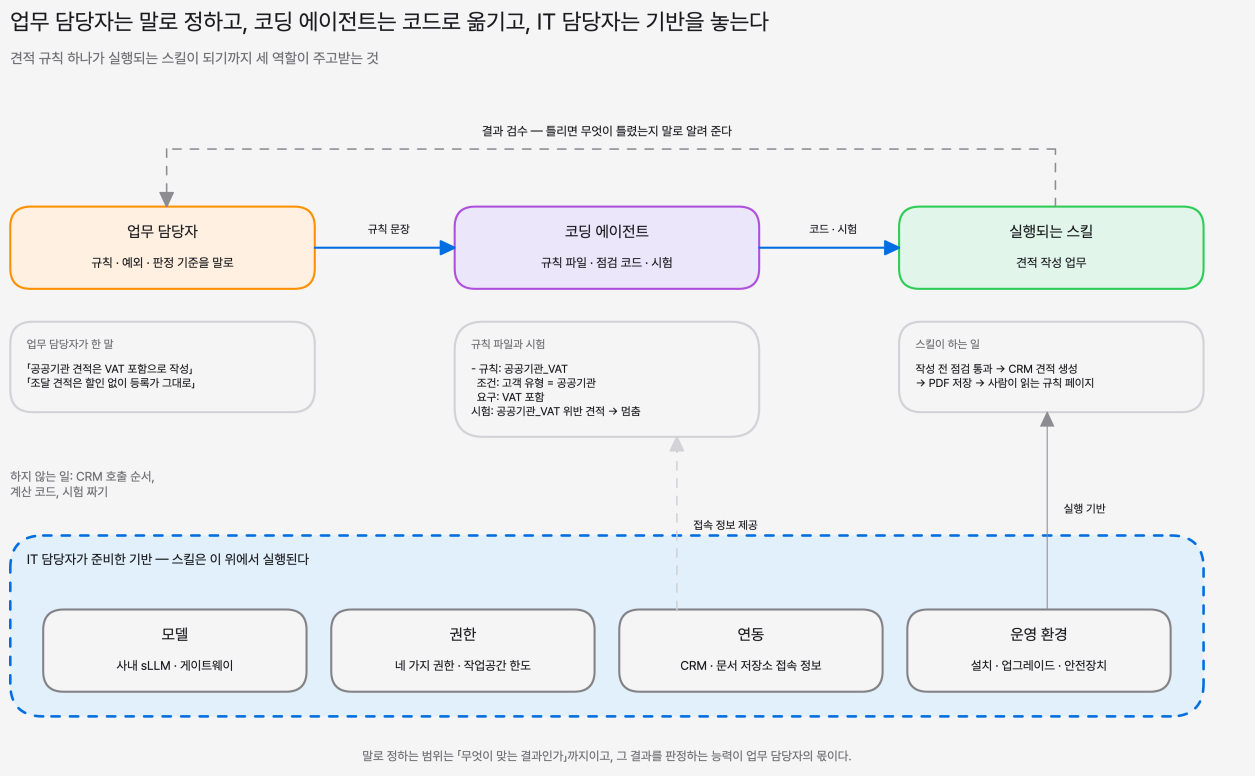


그림. 업무 담당자는 규칙과 판정 기준을 말로 정하고, 코딩 에이전트는 규칙 파일·점검 코드·시험을 만들고, IT 담당자는 모델·권한·시스템 연동을 맡는다

그림은 세 역할이 무엇을 주고받는지를 보여 줍니다. 업무 담당자가 말한 규칙 문장은 코딩 에이전트에게 넘어가 코드와 시험이 되고, 그 코드가 실행되는 모델과 연동은 IT 담당자가 준비한 기반 위에 있습니다.

5.3.1 업무 담당자가 말로 정하는 규칙·예외·판정 기준

말로 정할 수 있는 항목의 예

업무 담당자가 말로 정하는 것은 규칙, 예외, 판정 기준입니다. 견적 업무를 맡은 영업 업무 담당자가 평범한 문장으로 적은 요구에는 「공공기관 견적서는 VAT 포함으로 작성」, 「조달 제품의 경우에는 D/C 없이 조달가 그대로 작성」 같은 규칙이 들어 있었습니다. 같은 요구에는 「기존의 견적서를 확인하여 암묵지의 작성 원칙이 있는지 확인하여 반영」하라는 지시도 있었는데, 이 문장은 규칙을 찾는 방법을 정한 것입니다. 업무 담당자가 말로 정할 수 있는 항목은 다음 네 갈래로 나뉩니다.

갈래	업무 담당자가 하는 말의 예
규칙	공공기관 견적은 VAT 포함으로 작성한다

갈래	업무 담당자가 하는 말의 예
예외	조달 판매 정산 견적은 커스터마이징 비용이 없어도 사유를 남기면 통과한다
판정 기준	조달 등록가는 VAT 포함가이므로 품목 단가는 등록가 ÷ 1.1이어야 한다
찾는 방법	기존 견적서에서 문서에 없던 작성 관행을 찾아 반영한다

말로 정하는 범위의 끝

말로 정하는 범위는 「무엇이 맞는 결과인가」까지입니다. CRM에 어떤 순서로 무엇을 호출할지, 금액 계산을 어떤 코드로 할지, 시험을 어떻게 짤지는 업무 담당자가 정하지 않습니다. 다만 업무 담당자는 결과를 보고 틀렸다고 말할 수 있어야 하며, 이 검수 능력이 없으면 에이전트가 낸 그럴듯한 오답을 걸러낼 수 없습니다. 따라서 첫 업무는 업무 담당자가 결과의 옳고 그름을 바로 판정할 수 있는 업무로 고르는 편이 안전합니다.

5.3.2 코딩 에이전트가 만드는 코드·시험과 IT 담당자가 맡는 시스템 연동

업무 담당자가 하지 않아도 되는 일

업무 담당자가 규칙 문장을 넘기면 코딩 에이전트가 규칙 파일, 점검 코드, 시험 코드, 사람이 읽는 규칙 페이지를 만듭니다. OPENMARU AAS 사용자 매뉴얼은 사용자 PC의 코딩 에이전트 Claude Code를 AAS에 연결해 스킬을 만들어 올리고 실제 AI 에이전트로 시험하는 기능을 별도 쪽으로 설명하며[S12], 폐쇄망 환경에서는 같은 역할을 사내 코딩 모델로 바꿔 맡길 수 있습니다. 시스템 연동은 IT 담당자의 몫으로 남습니다. 세 역할의 경계를 정리하면 다음 표와 같습니다.

일	업무 담당자	코딩 에이전트	IT 담당자
규칙·예외·판정 기준 정하기	맡음	문장을 규칙 파일로 정리	해당 없음
점검 코드·시험 작성	결과만 확인	맡음	해당 없음
사람이 읽는 규칙 페이지	읽고 맞는지 확인	규칙 파일에서 자동 생성	해당 없음
CRM·문서 저장소 연결	해당 없음	연결 코드 작성	접속 정보·권한 제공
모델·안전장치	해당 없음	해당 없음	맡음

분담이 과장을 막는 방식

이 분담표는 「누구나 에이전트를 만든다」는 과장을 피하는 기준이기도 합니다. 업무 담당자에게 요구하는 능력은 규칙을 문장으로 쓰고 결과를 판정하는 능력이며, 이 능력이 없는 업무에는 에이전트를 맡기기 어렵습니다. 이 능력이 있는 업무라면 개발 인력을 새로 뽑지 않아도 코딩 에이전트와 IT 담당자의 기반만으로 에이전트를 만들 수 있습니다. 즉 조직이 먼저 확인할 것은 개발자 채용 여부보다 판정 기준을 문장으로 쓸 수 있는 업무 담당자가 있는지입니다.

5.4 부서가 직접 업무 프로그램을 만들던 전례와 담당자가 바뀌어도 남는 스킬

업무 담당자가 업무용 프로그램을 직접 만드는 일은 처음이 아닙니다. 1990년대에는 폭스프로(FoxPro)와 디베이스(dBASE) 같은 부서용 데이터베이스 도구가 보급되면서 전산실 개발자가 아닌 현장 담당자가 자기 업무용 테이블, 입력 화면, 보고서를 직접 만들었습니다. 그 시기의 경험은 현업이 직접 만들 때 생기는 장점과 한계를 함께 보여 주므로, AI Agent 시기의 제작 방식을 판단하는 데 좋은 비교 대상이 됩니다.

5.4.1 1990년대 부서용 데이터베이스 도구 시기와 AI Agent 시기의 비교

부서용 데이터베이스 도구 시기의 장점과 한계

부서용 데이터베이스 도구가 보급된 시기에는 엔진이 깔려 있으면 현장에서 별의별 업무 프로그램이 나왔고, 담당자는 전산실 일정을 기다리지 않았습니다. 하지만 그 프로그램은 만든 사람의 머릿속 규칙을 코드에 직접 넣은 형태였기 때문에, 문서가 남지 않아 만든 사람이 떠나면 아무도 고칠 수 없는 경우가 많았습니다. 반면 AI Agent 시기에는 업무 담당자가 말로 정한 규칙이 규칙 파일과 사람이 읽는 페이지로 남고, 코드는 그 규칙 파일에서 만들어집니다. 두 시기를 같은 기준으로 놓으면 다음과 같습니다.

기준	부서용 데이터베이스 도구 시기	AI Agent 시기(OPENMARU AAS)
기반	데이터베이스 엔진	에이전트 플랫폼과 사내 모델
업무 담당자가 만드는 것	테이블·화면·보고서	에이전트·스킬·예약 작업
규칙이 남는 형태	만든 사람이 짠 코드 안	규칙 파일과 사람이 읽는 규칙 페이지
고치는 방법	만든 사람이 직접 코드 수정	규칙 문장을 고치면 코딩 에이전트가 코드·시험 갱신
나누는 방법	디스켓·공유 폴더	사내 스킬 허브

AI Agent 시기에 달라진 유지 보수 경로

가장 큰 차이는 유지 보수 경로입니다. 부서용 데이터베이스 도구 시기의 프로그램은 코드가 곧 규칙이었으므로 규칙을 바꾸려면 코드를 읽을 줄 알아야 했습니다. 이에 비해 OPENMARU AAS에서 만드는 스킬은 규칙 파일이 정본이고 코드와 시험은 그 파일에서 파생되므로, 업무 담당자는 규칙 문장만 고치면 됩니다. 현업이 직접 만들던 시기의 장점인 속도는 유지하면서 한계였던 유지 보수 문제를 줄일 수 있습니다.

5.4.2 개인 노하우가 스킬·규칙 파일로 남는 조직 지식

스킬과 규칙 파일로 남는 업무 지식

OPENMARU AAS에서는 사용자마다 AI 에이전트 작업공간이 하나씩 만들어지고, 여기에 설정, 스킬, 기억, 파일이 저장됩니다[S12]. 업무 담당자가 말로 정한 규칙은 이 작업공간의 스킬과 규칙 파일로 남고, 같은 파일에서 사람이 읽는 규칙 페이지가 만들어집니다. 담당자가 부서를 옮기거나 회사를 떠나도 후임자는 규칙 페이지를 읽고 같은 스킬을 이어 쓸 수 있습니다. 개인의 노하우가 조직 지식으로 바뀌는 경로를 정리하면 다음 네 단계입니다.

1. 업무 담당자가 규칙과 예외를 문장으로 말한다
2. 코딩 에이전트가 문장을 규칙 파일, 점검 코드, 시험으로 옮긴다
3. 규칙 파일에서 사람이 읽는 규칙 페이지가 자동으로 만들어진다

4. 검증한 스킬을 사내 스킬 허브에 게시해 다른 부서가 설치해 쓴다

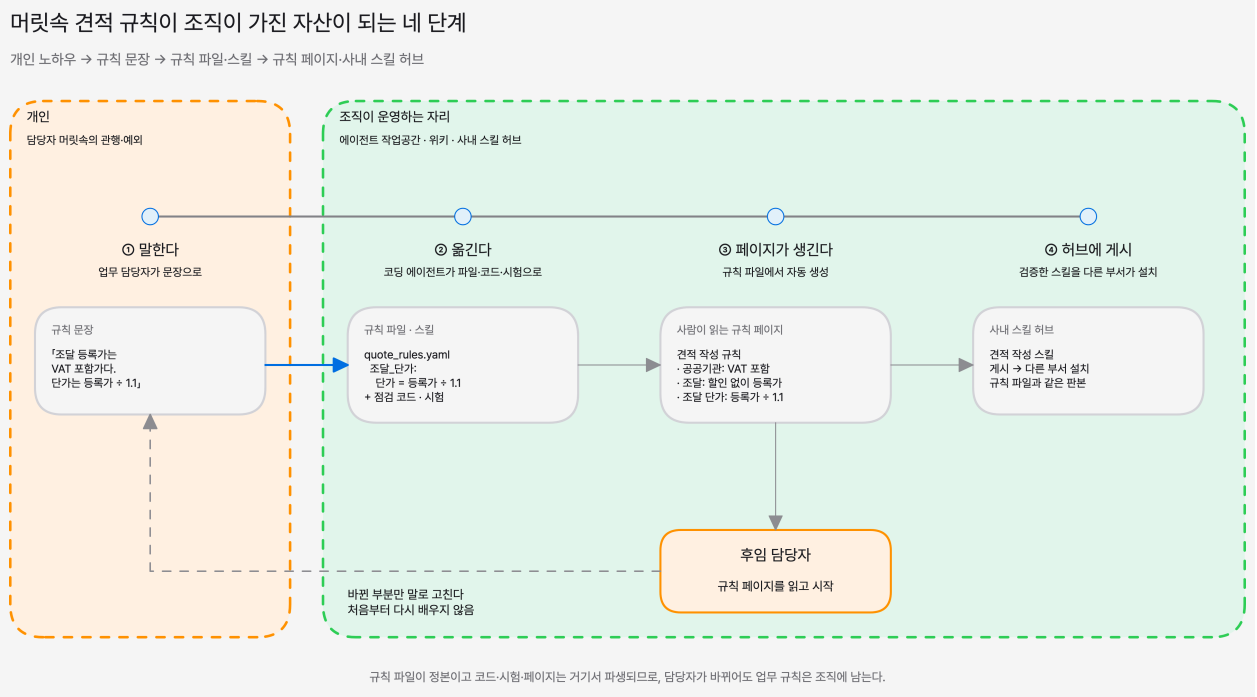


그림. 개인 노하우가 규칙 문장 → 규칙 파일·스킬 → 규칙 페이지·사내 스킬 허브를 거쳐 조직 자산이 되는 경로

담당자 교체 뒤에도 이어지는 업무

이 경로의 효과는 담당자 교체 때 가장 분명하게 드러납니다. 견적 규칙을 정한 영업 업무 담당자가 바뀌어도 규칙 파일과 규칙 페이지가 남아 있으므로, 후임자는 규칙을 처음부터 다시 배우지 않고 기존 규칙을 읽은 뒤 바뀐 부분만 말로 고치면 됩니다. 결국 업무 담당자가 AI Agent를 만드는 구조는 제작 속도를 높이는 방법인 동시에, 사람에게 묶여 있던 업무 규칙을 조직이 보유하는 자산으로 바꾸는 방법입니다.

조직이 운영하는 자리에 남는 에이전트 구성

담당자가 바뀌어도 업무가 이어지려면 에이전트를 이루는 요소가 개인 노트북이 아니라 조직이 운영하는 자리에 있어야 합니다. 에이전트 구성은 성격과 모델을 정한 프로파일, 일하는 방법을 적은 스킬, 실행 일정인 예약 작업, 그리고 회의록·보고서·고객 일지 같은 결과물의 네 가지로 볼 수 있습니다. 앞의 세 가지는 조직이 운영하는 에이전트 플랫폼에 저장되고, 결과물은 위키·CRM·문서 저장소처럼 조직이 이미 쓰는 시스템에 쌓입니다. 1990년대 부서용 데이터베이스 도구로 만든 업무 프로그램은 만든 사람이 떠난 뒤에도 파일로 남아 계속 쓰인 경우가 많았습니다. 이 네 가지가 조직의 자리에 있으면 에이전트도 사람보다 오래 남는 업무 자산이 됩니다.

6장: 견적 작성 사례로 보는 자연어 업무 정의와 에이전트 코딩

5장에서 나눈 분담이 실제 업무에서 어떻게 돌아가는지는 1장에서 결과물 업무의 대표로 든 견적 작성에서 가장 잘 드러납니다. 견적 작성은 OPENMARU AAS에서 업무 담당자가 말로 정의하고 에이전트가 매일 처리하는 대표 업무입니다. 견적서 한 장에는 공공기관 여부에 따른 VAT 표기, 조달 등록가 계산, 할인 표기, 제품별 추가 비용 같은 규칙이 겹쳐 있고, 이 규칙의 상당수는 문서가 아니라 영업 담당자들의 습관 속에 있습니다. 파이썬도 node.js도 모르는 업무 담당자가 이런 규칙을 에이전트에게 어떻게 가르치는지는 AI Agent 도입을 검토하는 조직이 가장 먼저 확인하고 싶어 하는 내용입니다. 견적 작성 업무는 두 단계로 나뉩니다. 만드는 단계에서는 영업 업무 담당자가 규칙을 평범한 문장으로 말하고 코딩 에이전트가 그 문장을 규칙 파일, 점검 코드, 시험으로 옮기며, 일하는 단계에서는 사내 sLLM 에이전트가 채팅으로 들어온 「견적」 요청을 받아 CRM 견적 생성부터 PDF 저장, 영업기회 일지 기록까지 처리합니다. 두 단계의 처리 과정을 차례로 따라가면 우리 조직의 어떤 업무를 같은 방식으로 맡길 수 있을지 판별할 기준도 함께 얻을 수 있습니다.

6.1 만드는 단계와 일하는 단계의 분리 — 코딩 에이전트와 사내 sLLM 에이전트의 역할

OPENMARU AAS로 업무 에이전트를 만들 때는 만드는 단계와 일하는 단계를 나눠 봐야 합니다. 두 단계는 일하는 주체, 다루는 자료, 자료가 흐르는 경로가 서로 다르기 때문에 보안 검토의 답도 단계마다 달라집니다. 만드는 단계는 규칙과 예시 자료를 다루며 규칙이 바뀔 때만 실행되고, 일하는 단계는 실제 고객 정보와 금액을 다루며 매일 실행됩니다. 따라서 「AI에 고객 자료를 보내도 되는가」라는 질문은 어느 단계의 이야기인지부터 정해야 정확히 답할 수 있습니다.

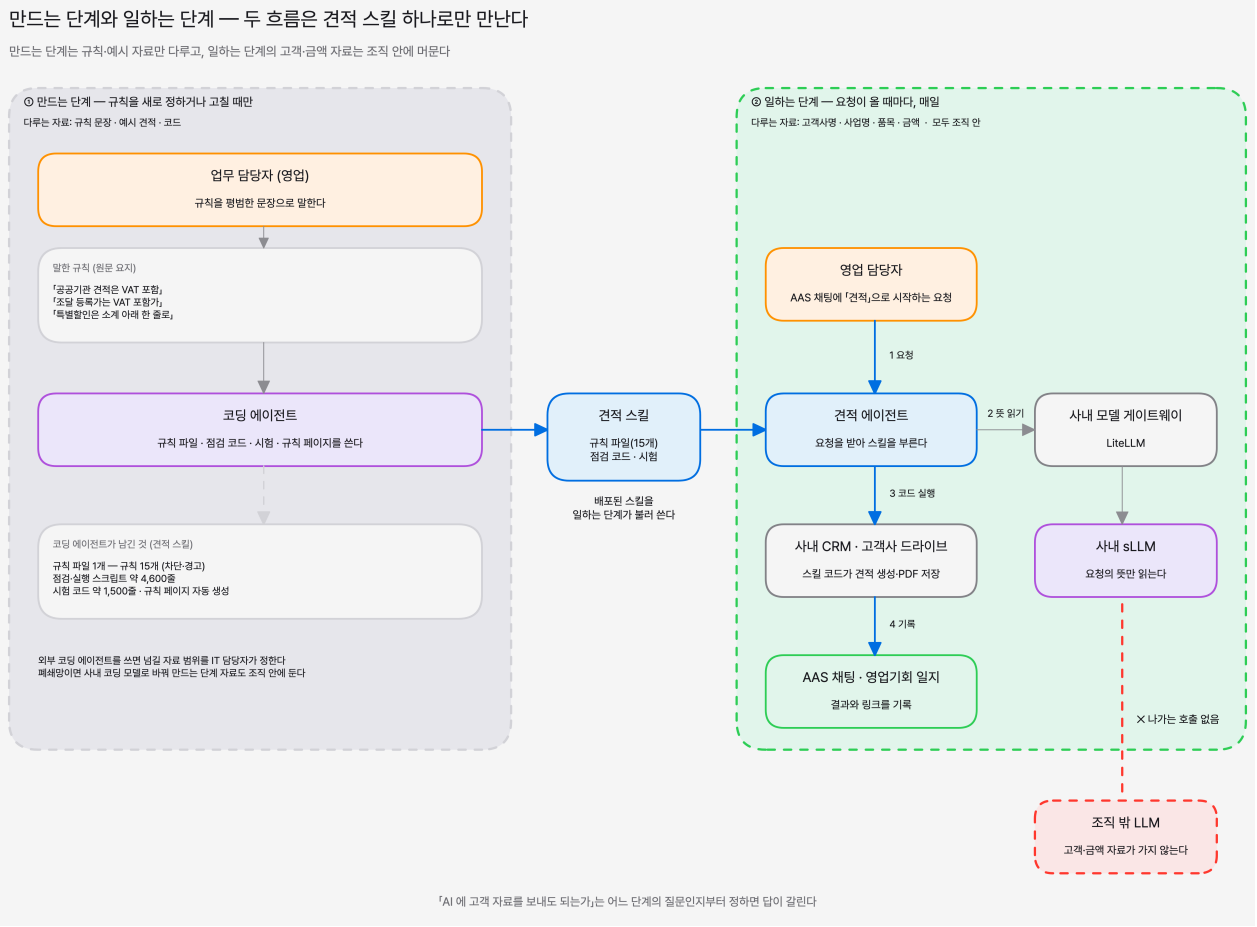


그림. 만드는 단계(업무 담당자의 말 + 코딩 에이전트, 규칙·예시 자료)와 일하는 단계(사내 sLLM 에이전트, 고객·금액 자료가 조직 안에 머무는 경로) — 두 흐름은 견적 스킬 하나로만 만난다

그림의 왼쪽은 규칙이 코드로 바뀌는 흐름이고 오른쪽은 매일의 견적 요청이 처리되는 흐름입니다. 두 흐름이 만나는 것은 코딩 에이전트가 만든 스킬 하나뿐이며, 오른쪽 흐름의 고객·금액 자료는 사내 모델 게이트웨이 밖으로 나가지 않습니다.

구분	만드는 단계	일하는 단계
누가	업무 담당자(말) + 코딩 에이전트	사내 sLLM 에이전트
무엇을	규칙 문장을 규칙 파일·점검 코드·시험·규칙 페이지로 옮김	「견적」 요청을 받아 확인·CRM 견적 생성·PDF 저장·일지 기록
언제	규칙을 새로 정하거나 고칠 때	영업 담당자가 요청할 때마다
다루는 자료	규칙 문장, 예시 견적, 코드	고객사명, 사업명, 품목, 금액
자료 흐름	연결한 코딩 에이전트까지	사내 모델 게이트웨이 안에서만

6.1.1 규칙·예시 자료만 다루는 만드는 단계와 폐쇄망의 사내 코딩 모델 대체

코딩 에이전트 연결로 진행하는 만드는 단계

만드는 단계에서 규칙 문장을 코드로 옮기는 일은 코딩 에이전트가 합니다. OPENMARU AAS 사용자 매뉴얼은 사용자 PC에서 실행되는 코딩 에이전트 Claude Code를 AAS에 연결해 스킬을 만들어 올리고 실제 AI 에이전트로 시험하는 「Claude Code 연결」 기능을 따로 설명하고, 관리자 매뉴얼은 모든 사람의 연결을 한 화면에서 보고 강제로 끊는 관리 항목으로 다룹니다[S12][S13]. 현재 견적 스킬은 외부 코딩 에이전트인 Claude Code로 만들었습니다. 코딩 에이전트가 맡는 일은 규칙 파일 작성, 작성 진행기와 점검 코드 작성, CRM 연결 코드 작성, 시험 코드 작성, 사람이 읽는 규칙 페이지 생성입니다.

만드는 단계에서 다루는 자료 범위

만드는 단계에서 코딩 에이전트가 보는 자료는 규칙 문장, 예시 견적, 그리고 자신이 작성하는 코드입니다. 다만 지난 견적에서 문서 밖 관행을 찾는 작업처럼 실제 견적 자료를 읽어야 하는 작업도 있으므로, 외부 코딩 에이전트를 쓸 때는 어떤 자료를 넘길지 IT 담당자가 미리 정해야 합니다. 자료를 조직 밖으로 보낼 수 없는 폐쇄망 조직은 같은 역할을 사내 코딩 모델로 바꿔 맡길 수 있으며, 이 경우 만드는 단계의 자료도 조직 안에 머뭅니다. 만드는 단계의 자료 범위를 정리하면 다음과 같습니다.

자료	만드는 단계에서의 쓰임	외부 코딩 에이전트 사용 시 고려
규칙 문장	규칙 파일의 원문	업무 규칙이므로 대개 공개 수준
예시 견적	규칙과 시험의 예시값	고객명·금액을 가린 예시로 대체 가능
지난 견적 전체	문서 밖 관행 찾기, 새 규칙 재점검	넘길 범위를 IT 담당자가 결정, 폐쇄망이면 사내 코딩 모델
코드·시험	코딩 에이전트의 산출물	고객 자료 미포함

6.1.2 고객·금액 자료가 조직 밖으로 나가지 않는 일하는 단계

사내 모델 게이트웨이를 거치는 자료 경로

일하는 단계에서 견적을 만드는 주체는 사내 sLLM 위에서 실행되는 에이전트입니다. OPENMARU AAS 관리자 매뉴얼에 따르면 AI 에이전트는 LiteLLM을 거쳐 AI 모델을 부르므로[S13], 모델 게이트웨이를 사내 sLLM에만 연결해 두면 견적 요청에 담긴 고객사명과 금액은 조직 밖 LLM으로 전달되지 않습니다. AAS는 OpenAI 호환 주소 하나로 사내 vLLM, LiteLLM, Ollama 같은 모델 서버에 연결하며, 도구를 쓰는 에이전트에는 64,000 토큰 이상의 문맥을 지원하는 모델이 필요합니다[S14]. 일하는 단계의 자료 경로는 다음 순서로 이어집니다.

1. 영업 담당자가 AAS 채팅에 「견적」으로 시작하는 요청을 쓴다
2. 에이전트가 사내 모델 게이트웨이를 거쳐 사내 sLLM으로 요청의 뜻을 읽는다
3. 미리 만든 견적 스킬의 코드가 사내 CRM과 고객사 드라이브 폴더에 접속해 견적을 만들고 저장한다
4. 결과와 링크가 다시 AAS 채팅과 영업기회 일지에 기록된다

보안 검토에 쓰는 두 단계 구분

이 경로에서 외부 서비스로 나가는 호출은 없습니다. 그래서 보안 검토는 두 질문으로 나눠 답할 수 있습니다. 매일 실행되는 일하는 단계에서는 고객·금액 자료가 사내 모델 게이트웨이 안에서만 처리된다는 점을 모델 게이트웨이 설정으로 확인하고, 가끔 실행되는 만드는 단계에서는 코딩 에이전트에 넘기는 자료 범위를 정하거나 사내

코딩 모델로 바꾸면 됩니다. 이 구분은 4장에서 다룬 sLLM 판단과 이 사례를 잇는 핵심이며, 외부 LLM이 더 뛰어나더라도 매일의 업무 자료를 내보내지 않고 에이전트를 운영할 수 있다는 근거가 됩니다.

6.2 공공기관 VAT 포함·조달 등록가 기준 등 업무 담당자가 말로 정한 견적 작성 규칙

견적 스킬의 출발점은 영업 업무 담당자가 평범한 문장으로 적은 요구입니다. 그 요구에는 「견적이라고 첫 번째 줄에 작성하면 자동으로 고객사와 사업명을 입력받는」 확인 절차, 「공공기관인데 VAT 별도로 견적이 작성되었다던지」 하는 이상을 확인하는 점검 루틴, 그리고 반드시 지켜야 할 규칙 몇 줄이 들어 있습니다. 문장이 다듬어지지 않고 맞춤법이 정확하지 않아도 코딩 에이전트가 규칙을 뽑아내기에는 충분합니다. 업무 담당자가 말로 정한 규칙을 정리하면 다음과 같습니다.

업무 담당자가 한 말(요지)	규칙 파일에 들어간 규칙	점검 결과
공공기관 견적서는 VAT 포함으로 작성	제목에 VAT 포함 말머리와 VAT 포함 서식, 세액 이중 부과 금지	위반 시 차단
조달 견적이면 물품식별번호를 품목 설명에	조달 품목 설명에 물품식별번호 한 줄	위반 시 차단
조달 제품은 D/C 없이 조달가 그대로	할인 0%, 정가 = 단가 = 등록가 기준 금액	위반 시 차단(정산 등 예외는 사유를 남기면 통과)
조달 등록가는 VAT 포함가	VAT 별도 견적의 품목 단가 = 등록가 ÷ 1.1	등록가를 그대로 넣으면 차단
특별할인은 품목 금액에 녹이지 말고 소계 아래 한 줄로	품목 금액은 그대로 두고 특별가 줄에 총액 표시	위반 시 차단
플랫폼 제품 판매 시 커스터마이징 비용 포함 여부 확인	플랫폼 제품 견적에 커스터마이징 품목 포함	위반 시 차단(예외는 사유를 남기면 통과)

6.2.1 지난 견적에서 AI가 찾아낸 문서 밖 작성 관행

지난 견적에서 관행을 찾으라는 지시

업무 담당자의 요구에는 「기존의 견적서를 확인하여 암묵지의 작성 원칙이 있는지 확인하여 반영」하라는 문장이 있었습니다. 이 지시에 따라 코딩 에이전트는 CRM에 쌓인 지난 견적 전체를 읽고, 문서로 정리된 적은 없지만 견적마다 되풀이되는 작성 방식을 찾았습니다. 사람이 수천 건의 견적을 한 장씩 넘겨 보며 공통점을 찾기는 어렵지만, 코딩 에이전트는 제목 형식, 세액 칸의 값, 소계와 총액의 관계, 조건 필드의 값을 견적 전체에서 세어 보고 반복되는 형태를 규칙 후보로 올립니다.

VAT 포함 표기 관행과 기본 조건 값

가장 대표적인 관행은 VAT 포함 여부의 표기 방식입니다. 공공기관 견적은 세액 칸에 금액을 적지 않고, 견적 제목 앞에 VAT 포함 말머리를 붙인 뒤 VAT 포함 전용 PDF 서식으로 출력해 왔습니다. 이 관행을 모르는 사람이 세액 칸에 VAT를 따로 적으면 VAT 포함 서식과 겹쳐 세금이 두 번 계산되므로, 규칙 파일은 이를 세액 이중 부과 금지 규칙으로 옮겼습니다. 또한 지불 조건, 납기, 유효기간, 설치 장소에 가장 자주 쓰인 문구를 찾아 새 견적의 기본값으로 넣었고, 이 네 칸이 비어 있으면 경고를 내도록 했습니다. 결과적으로 지난 견적에서 찾은 관행은

VAT 포함 말머리와 서식을 자동으로 고르는 기능, 세액 이중 부과를 막는 점검, 기본 조건을 채우는 기능의 세 가지로 견적 스킬에 들어갔습니다.

6.2.2 말로 고친 규칙 한 줄이 규칙 파일·점검 코드·시험·규칙 페이지로 바뀌는 흐름

업무 담당자가 말로 바로잡은 두 규칙

코딩 에이전트가 만든 규칙이 업무 담당자의 뜻과 다르면 업무 담당자는 말로 바로잡습니다. 예를 들어 특별할인이 품목마다 할인율로 나뉘어 들어간 견적을 보고 업무 담당자가 「특별할인은 품목 금액에 녹이지 말고 소계 아래 한 줄로」라고 말하면, 코딩 에이전트는 이 한 문장을 받아 품목 금액은 정가대로 두고 소계 아래 특별가 줄에 총액을 표시하는 규칙과, 특별가 견적의 비고에 해당 사업에만 적용되는 가격이라는 문구를 넣는 규칙을 함께 추가합니다. 또 업무 담당자가 「조달 등록가는 VAT 포함가」라고 정하면, VAT 별도 견적의 품목 단가를 등록가 ÷ 1.1로 계산하도록 반영합니다. 예를 들어 등록가가 11,000,000원인 품목은 공급가 10,000,000원으로 들어가고, VAT를 더한 금액이 등록가와 같아집니다.

규칙 한 줄이 네 가지 산출물로 바뀌는 흐름

업무 담당자는 규칙을 말로만 고치고, 규칙 파일·점검 코드·시험·사람이 읽는 규칙 페이지는 코딩 에이전트가 만듭니다. 견적 스킬의 규칙은 규칙 15개를 담은 규칙 파일 하나에 모여 있고, 이 규칙을 실행하는 스크립트 약 4,600줄과 시험 코드 약 1,500줄도 코딩 에이전트가 작성한 것입니다. 사람이 읽는 규칙 페이지도 같은 규칙 파일에서 자동으로 만들어지므로, 규칙 파일과 페이지의 내용이 서로 어긋나지 않습니다.

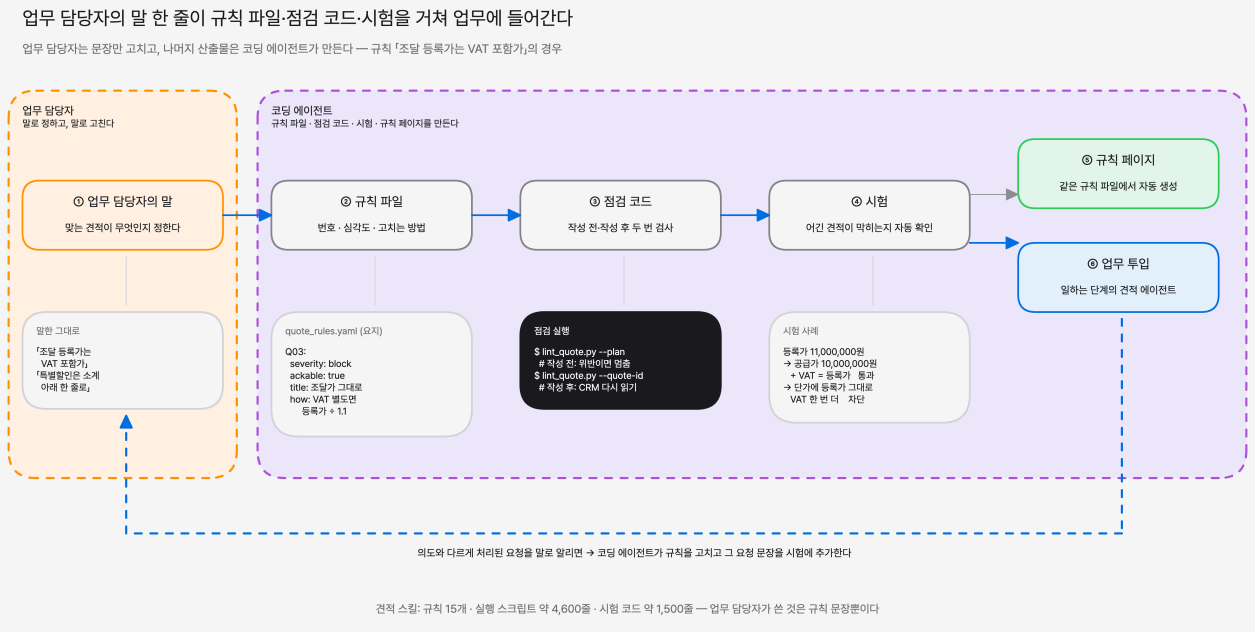


그림. 업무 담당자의 말 → 규칙 파일 → 점검 코드 → 시험 → 규칙 페이지·업무 투입으로 이어지는 흐름과 말로 고치는 개선 순환

산출물	만드는 주체	하는 일
규칙 문장	업무 담당자	무엇이 맞는 견적인지 정함
규칙 파일	코딩 에이전트	규칙마다 번호·심각도(차단·경고)·고치는 방법을 기록

산출물	만드는 주체	하는 일
점검 코드	코딩 에이전트	규칙 파일을 읽어 견적을 작성 전후로 검사
시험 코드	코딩 에이전트	규칙을 어긴 견적이 실제로 막히는지 자동 확인
규칙 페이지	코딩 에이전트(자동 생성)	영업 담당자가 읽는 규칙 설명서

6.3 「견적」 요청에서 CRM 등록·PDF 저장·영업기회 일지 기록까지의 처리 9단계

일하는 단계에서 견적 에이전트는 채팅 첫 줄이 「견적」인 요청을 받아 아홉 단계로 처리합니다. 의사결정권자가 이 업무를 에이전트에게 맡길 수 있는지 판단하려면 사람이 확인하는 단계가 어디에 있고, 에이전트가 혼자 진행하는 단계가 어디까지인지 알아야 합니다. 견적 처리에서 사람이 확인하는 단계는 네 번이며, CRM에 견적을 실제로 만드는 일은 마지막 확인을 받은 뒤에만 실행됩니다. 아홉 단계를 사람 확인 여부와 함께 정리하면 다음과 같습니다.

단계	에이전트가 하는 일	사람 확인
1	고객사, 사업명, 수신처, 공공기관 여부, 조달 여부, 견적 유형을 한 번에 묻는다	확인 ①
2	CRM과 영업기회 페이지에서 기존 영업기회 후보를 찾아 보여 주고, 없으면 새로 등록한다	확인 ②
3	영업기회 아래에 견적 요청 페이지(필수 확인 항목·품목 표·조건·비고)를 만든다	—
4	품목, 조건, 비고를 확정받는다	확인 ③
5	작성 전 점검을 실행하고 차단 규칙 위반이 있으면 멈춘다	—
6	제목, 금액, 경고를 보여 주고 CRM에 만들지 최종 확인을 받는다	확인 ④
7	CRM(SuiteCRM)에 견적과 품목을 만들고 견적번호를 제목에 넣는다	—
8	CRM을 다시 읽어 작성 후 점검을 하고, PDF를 고객사 드라이브 폴더의 견적서 폴더에 저장한다	—

단계	에이전트가 하는 일	사람 확인
9	견적 요청 페이지와 영업기회 일지에 CRM 견적·PDF·폴더 링크를 기록하고 완료를 알린다	—

6.3.1 만들기 전과 만든 뒤 두 번 점검하고 규칙 위반 견적은 만들기 전에 멈추는 구조

작성 전 점검과 작성 후 점검의 역할

견적 에이전트는 같은 규칙 파일로 두 번 점검합니다. 작성 전 점검은 확정된 품목과 조건을 CRM에 보내기 전에 검사해, 차단 규칙을 어긴 견적은 아예 만들지 않습니다. 작성 후 점검은 CRM에 만들어진 견적을 다시 읽어 저장 과정에서 금액이나 서식이 바뀌지 않았는지 확인합니다. 예를 들어 VAT 포함 견적인데 CRM의 총액이 소계에 1.1을 곱한 값으로 저장돼 있다면 품목에 공급가가 들어간 것이므로, 작성 후 점검이 이를 잡아 요청 페이지에 표시합니다.

작성 전에 멈추는 조건과 경고 조건

점검 결과는 차단과 경고 두 가지로 나뉩니다. 차단은 견적을 만들지 않고 멈추는 결과이고, 경고는 내용을 보여준 뒤 사람이 확인하면 진행하는 결과입니다. 공공기관인데 VAT 별도로 작성된 견적, 조달 견적에 할인이 걸린 견적은 대표적인 차단 조건입니다. 주요 조건을 정리하면 다음과 같습니다.

조건	판정	진행 방법
공공기관 견적인데 VAT 별도	차단	VAT 포함으로 고친 뒤 다시 점검
조달 견적에 할인율 적용	차단	할인을 빼고 등록가 기준으로 수정, 정산 등 예외는 사유 기록
조달 품목 단가에 등록가를 그대로 입력	차단	등록가 ÷ 1.1로 수정
특별할인을 품목 금액에 반영	차단	품목은 정가로 두고 소계 아래 특별가 줄로 이동
플랫폼 제품 견적에 커스터마이징 비용 없음	차단	품목 추가 또는 예외 사유 기록
비고가 비어 있음	차단	제품별 표준 비고 문구 추가
할인 90% 이상 또는 단가 계산 불일치	경고	사람이 확인한 뒤 진행
지불·납기·유효기간·설치 장소 빈칸	경고	사람이 확인한 뒤 진행

6.3.2 요청 뜻은 모델이 읽고 판단은 미리 만든 코드가 맡는 처리 구조

작은 모델에 맞춘 역할 나누기

사내 sLLM 같은 작은 모델은 짧은 문장의 뜻을 가르는 일은 잘하지만, 여러 단계를 이어 가며 도구를 고르고 결과를 판단하는 긴 작업에서는 흔들리기 쉽습니다. 그래서 견적 에이전트는 「판단은 코드가, 요청 해석은 모델이」

라는 원칙으로 역할을 나눕니다. 모델은 「견적 PDF 받아 줘」, 「지난 견적 찾아 줘」, 「품목 하나 빼 줘」 같은 요청의 뜻을 읽어 어떤 처리인지 고르고, 그 뒤의 CRM 조회, 금액 계산, 규칙 점검, 파일 저장은 미리 만든 코드가 실행합니다. Anthropic도 공식 글에서 정해 둔 코드 경로로 LLM과 도구를 엮는 방식을 워크플로라 부르고, 가장 단순한 해법부터 쓰고 필요할 때만 복잡도를 올리라고 권합니다[S11].

일	모델이 맡는 몫	코드가 맡는 몫
요청 해석	요청이 새 견적·PDF 받기·찾기·수정 가운데 무엇인지 고름	고른 결과가 규칙에 맞는지 검증
값 추출	문장에서 고객사명·견적번호·품목을 뽑음	CRM에서 실제 존재 여부 확인
금액 계산	관여하지 않음	VAT·등록가·할인 계산
규칙 판단	관여하지 않음	규칙 파일로 차단·경고 판정
결과 전달	코드가 낸 결과를 사람이 읽기 좋은 문장으로 전달	링크·점검표 작성

도구 직접 호출 없이 끝나는 견적 요청

이 구조에서는 모델이 도구를 직접 부르지 않습니다. OPENMARU AAS 채팅에서 견적 PDF 요청은 10초 안팎, 지난 견적 찾기는 7초 안팎에 처리되며, 이 과정에서 모델은 요청의 뜻을 고르는 일만 하고 CRM 조회·금액 계산·규칙 점검은 미리 만든 코드가 실행합니다. 즉 작은 모델로 업무 에이전트를 운영하는 조건은 판단할 일을 코드로 미리 만들어 두는 것입니다.

6.3.3 규칙 변경 때 지난 견적 전체를 다시 대어 보는 통과·경고·차단 점검

새 규칙으로 지난 견적을 다시 점검하는 기능

규칙을 바꾸면 새 규칙을 지난 견적 전체에 다시 대어 볼 수 있습니다. 점검 코드는 새 견적을 검사할 때와 같은 규칙 파일로 지난 견적을 한 건씩 읽어 통과, 경고, 차단으로 가르고, 규칙을 바꾸기 전후로 몇 건이 어느 판정으로 옮겨 갔는지 바로 보여 줍니다. 예를 들어 조달 등록가가 VAT 포함가라는 규칙을 넣고 재점검하면, 등록가를 그대로 단가로 넣어 VAT가 한 번 더 붙은 지난 견적이 차단으로 표시됩니다. 업무 담당자는 이 목록을 보고 규칙이 너무 엄격한지, 아니면 실제로 고쳐야 할 견적이 있는지를 판단합니다.

재점검 결과 표의 구성

재점검 결과는 규칙마다 한 줄씩 나오는 표로 정리됩니다. 각 줄에는 규칙 번호와 제목, 통과·경고·차단으로 판정된 견적 수, 판정별 대표 견적의 번호와 제목이 함께 표시되므로, 업무 담당자는 숫자만 보지 않고 실제 견적을 열어 판정이 맞는지 확인할 수 있습니다. 마지막 열에는 규칙 수정, 예외 사유 추가, 지난 견적 정정 가운데 어떤 조치를 할지 적습니다. 이 표는 규칙을 확정하기 전에 영향을 미리 확인하는 용도로 쓰이며, 조달 판매 정산 견적처럼 예외 사유를 남겨야 할 견적 유형도 이 표에서 드러납니다.

6.4 견적 작성 다음 후보 업무의 판별 기준 — 문장으로 정리되는 규칙과 반복 빈도

견적 작성이 첫 업무로 적합했던 이유는 규칙을 문장으로 쓸 수 있고, 같은 형태의 요청이 자주 들어오며, 결과가 맞는지 업무 담당자가 바로 판정할 수 있었기 때문입니다. 같은 방식으로 OPENMARU AAS에서는 영업기회

등록, 기술지원 확약서 작성, 이력서 직군별 정리, 마케팅 지표 질의응답 같은 업무도 에이전트가 처리하고 있습니다. 다음 업무를 고를 때도 같은 기준을 쓰면 시행착오를 줄일 수 있으며, 첫 시범 도입 대상은 이 기준에서 높은 점수를 받는 업무 하나로 정하는 편이 안전합니다.

6.4.1 규칙 명확성·반복 빈도·연결 시스템 수·자료 민감도의 첫 업무 판별표

첫 업무 판별표의 3점 척도 채점

첫 업무 후보는 규칙 명확성, 반복 빈도, 연결 시스템 수, 자료 민감도의 네 기준으로 3점 척도를 매겨 고릅니다. 규칙 명확성은 업무 담당자가 맞는 결과의 조건을 문장으로 쓸 수 있는지를 보고, 반복 빈도는 같은 형태의 요청이 얼마나 자주 들어오는지를 봅니다. 연결 시스템 수는 적을수록 IT 담당자의 연동 부담이 작으므로 점수가 높고, 자료 민감도는 높을수록 사내 sLLM으로 처리할 이유가 커지므로 점수를 높게 줍니다. 견적 작성은 연결 시스템이 CRM, 영업기회 페이지, 드라이브 세 곳이라 연동 점수가 중간이지만, 나머지 세 기준에서 높은 점수를 받습니다.

기준	3점	2점	1점	견적 작성 예시 점수
규칙 명확성	맞는 결과의 조건을 문장으로 모두 쓸 수 있음	대부분 쓸 수 있고 일부는 사람 판단	담당자마다 판단이 다름	3
반복 빈도	매일 여러 번	매주 여러 번	한 달에 몇 번	3
연결 시스템 수	1곳	2–3곳	4곳 이상	2
자료 민감도	고객·금액·개인정보 포함	내부 자료	공개 자료	3

판별표 점수의 해석

합계가 높은 업무부터 시범 도입 대상으로 검토하되, 규칙 명확성이 1점인 업무는 합계와 관계없이 뒤로 미룹니다. 판정 기준을 문장으로 쓰지 못하는 업무는 에이전트가 낸 결과를 업무 담당자도 판정할 수 없기 때문입니다. 반면 연결 시스템 수가 많은 업무는 IT 담당자가 연동을 먼저 준비하면 점수를 올릴 수 있으므로, 연동 계획과 함께 두 번째 업무로 검토할 수 있습니다.

기능 목록이 아니라 업무 목록에서 출발하는 판별

판별표에 올릴 후보는 에이전트 플랫폼의 기능 목록이 아니라 부서의 업무 목록에서 고릅니다. 에이전트 플랫폼의 문서는 메신저 연결, 예약 실행, 스킬, 작업 보드, 하위 에이전트 위임을 서로 독립된 구성 요소로 나눠 설명합니다. 반면 조직이 인식하는 단위는 「주간 운영 보고」나 「견적 작성」처럼 여러 기능이 한 업무 안에 함께 들어간 형태입니다. 기능 목록에서 출발하면 「예약 실행을 어디에 쓸까」라는 질문에 머뭅니다. 업무 목록에서 출발하면 그 업무 하나를 끝내는 데 어떤 기능이 몇 개 필요한지가 드러납니다. 판별표를 채우기 전에 부서가 반복하는 업무를 먼저 적고, 업무마다 필요한 기능을 대응시키는 순서가 판단을 쉽게 만듭니다.

6.4.2 말로 알려 준 오류가 에이전트 수정으로 이어지는 개선 순환

처음 의도와 다르게 처리된 요청의 수정 사례

업무 에이전트는 처음부터 완성된 상태로 시작하지 않습니다. 예를 들어 첫 줄이 「견적」이면 새 견적을 시작한다는 규칙만 있으면 「견적 ○○○○ PDF 다운로드해」 같은 요청도 새 견적 작성으로 처리될 수 있습니다. 영업 담당자가 이 결과를 말로 알려 주면 코딩 에이전트는 첫 줄만 보지 않고 요청의 뜻을 먼저 읽는 단계를 더해 PDF 받기, 견적 찾기, 수정, 새 견적 작성을 가르도록 고칩니다. 그리고 잘못 처리됐던 요청 문장을 시험 코드에 추가해 같은 오류가 다시 생기지 않는지 계속 확인합니다.

개선 순환의 네 단계

이 수정 과정은 AI 에이전트가 행동하고 결과를 확인하며 계획을 고쳐 가는 반복 구조와 같은 모양을 조직 수준에서 갖춘 것입니다[S10]. 업무 담당자와 영업 담당자는 코드를 보지 않고 말로만 오류를 알리며, 순환은 다음 네 단계로 이어집니다.

1. 업무 담당자나 사용자가 처음 의도와 다르게 처리된 요청을 말로 알린다
2. 코딩 에이전트가 원인을 찾아 규칙 파일이나 처리 단계를 고친다
3. 오류를 낸 요청 문장을 시험 코드에 추가해 같은 오류가 다시 생기지 않는지 확인한다
4. 고친 스킬을 다시 배포하고, 바뀐 규칙은 규칙 페이지에 자동으로 반영된다

결국 업무 담당자가 AI Agent에게 일하는 방법을 가르치는 과정은 한 번의 개발 프로젝트로 끝나지 않고, 말로 알려 주고 고쳐지는 순환으로 이어집니다. 이 순환이 짧을수록 에이전트는 업무 담당자가 생각하는 맞는 견적에 빨리 가까워지며, 그 속도를 정하는 것은 코딩 실력보다 무엇이 틀렸는지 문장으로 말할 수 있는 업무 담당자의 판정 능력입니다.

7장: OPENMARU AAS를 만든 배경과 제품 정의

6장의 견적 작성 사례는 만들기·시키기·지켜보기가 한 흐름으로 이어질 때 업무 담당자가 에이전트를 직접 가르칠 수 있다는 점을 보여 줍니다. 그 흐름을 담는 도구가 OPENMARU AAS입니다. OPENMARU AAS(AI Agent Studio)는 오픈소스 에이전트 플랫폼 Hermes Agent에 연결해, 업무 담당자가 자기 AI 에이전트를 만들고 일을 시키고 그 결과를 지켜보는 웹 Studio입니다. 코드는 공개 저장소 openmaru-io/ai-agent-studio에 Apache-2.0 라이선스로 공개돼 있고, 외부 서비스에 가입하지 않고 조직 안에 설치합니다[S14]. 에이전트 플랫폼은 에이전트를 실행하는 엔진이므로 엔진이 안정적이어도 사람이 에이전트와 일하는 창구가 메신저 채팅 하나뿐이면 업무에 정착하기 어렵습니다. 2장에서 본 Gartner 전망(2027년 말까지 에이전틱 AI 프로젝트의 40% 이상 취소)이 꼽은 세 원인 가운데[S4], 무엇이 실패했는지 늦게 아는 운영 상태는 위험 통제 부족에 해당합니다. 채팅 창구가 남기는 운영 공백을 짚어 보면, 그 공백에서 나온 제품 정의와 기존 플랫폼 대시보드와 나누는 역할도 함께 드러납니다.

7.1 에이전트 플랫폼의 채팅 창구가 남긴 세 가지 운영 공백

AI Agent 플랫폼은 대개 메신저 채팅을 기본 창구로 제공합니다. Hermes Agent도 텔레그램, 디스코드, 슬랙, 매터모스트 같은 메신저를 연결해 에이전트와 대화하게 합니다[S14]. 채팅 창구는 질문을 보내고 답을 받는 데는 충분하지만, 에이전트에게 실제 업무를 맡기기 시작하면 세 가지 일이 채팅 화면에서 처리되지 않습니다. 에이전트와 스킬을 만드는 일, 맡긴 일이 제대로 끝났는지 확인하는 일, 잘 만든 에이전트와 스킬을 조직 안에서 나누는 일입니다. 에이전트의 정의가 「환경의 피드백을 보며 도구를 쓰는 반복」이라면[S11], 그 반복이 지금 어디까지 왔는지를 사람이 볼 수 있어야 업무를 맡길 수 있습니다.

7.1.1 만들기·맡긴 일의 확인·조직 내 공유를 담지 못하는 채팅 창구

에이전트 구성과 분리된 대화 창

채팅 창에서는 지금 대화하는 에이전트가 어떤 모델을 쓰는지, 어떤 스킬을 갖고 있는지, 정해진 시각에 무엇을 실행하는지로 바로 넘어갈 수 없습니다. 그 정보는 설정 파일, 관리 대시보드, 명령줄에 나뉘어 있기 때문입니다. 에이전트를 새로 만들거나 일하는 방법을 고치려면 업무 담당자가 플랫폼을 다룰 줄 아는 사람에게 작업을 요청해야 합니다. 메신저 봇도 에이전트마다 등록하고 토큰을 복사해 채널을 연결해야 하므로, 에이전트를 하나 더 만들 때마다 같은 설정 작업을 되풀이합니다. 결과적으로 대화와 구성이 서로 다른 도구에 흩어져 있어 만들기 단계가 업무 담당자의 손을 떠나게 됩니다.

맡긴 일의 진행·승인·실패 확인

메신저 채팅은 사람끼리의 대화를 위해 설계됐으므로 에이전트가 지금 어떤 단계를 밟는지, 어떤 명령을 실행하려 하는지, 무엇을 되묻는지를 따로 보여 주는 화면 요소가 없습니다. 승인이 필요한 명령도 메시지 한 줄로 지나가고, 긴 표와 코드는 읽기 어렵습니다. 예약 작업은 확인이 더 어렵습니다. 에이전트 플랫폼을 운영하면 흔히 생기는 문제는 실패 알림이 그 작업의 결과 전달 대상으로만 가도록 설정된 경우입니다. 이때 결과를 로컬 파일로만 남기는 예약 작업은 실패해도 아무에게도 알리지 않고, 실행이 멈춘 작업은 누군가 결과를 찾을 때까지 드러나지 않습니다. 알림 상태가 「보냄」으로 기록되더라도 사람이 실제로 받았다는 뜻은 아니므로, 채팅 창구만으로는 맡긴 일이 끝났는지 확인할 방법이 없습니다.

한 사람의 에이전트에 머무는 스킬

잘 만든 스킬은 처음 만든 사람의 에이전트 안에만 있는 경우가 많습니다. 그 스킬이 어떤 MCP(Model Context Protocol) 서버, REST API, 스크립트를 부르는지는 스킬 문서와 스크립트 파일을 직접 열어 읽어야 알 수 있습니다. 채팅 창구에는 다른 부서가 그 스킬을 찾아 설치하거나, 사람과 여러 에이전트가 한 과제를 나눠 맡는 기능이 없습니다. 그래서 업무 노하우가 담당자 개인의 환경에 남고, 담당자가 바뀌면 함께 사라질 위험이 생깁니다.

세 가지 공백을 업무 장면별로 나누면 다음 표와 같습니다. 오른쪽 열은 OPENMARU AAS가 각 공백에 대응하는 현재 기능이며, 모두 사용자 매뉴얼에 설명된 기능입니다[S12].

공백	채팅 창구에서 생기는 일	OPENMARU AAS의 대응
만들기	모델·스킬·예약 작업 설정이 파일과 명령줄에 나뉘어 있고, 봇 연결을 에이전트마다 반복	프로파일 화면에서 성격·모델·스킬·MCP 연결·예약 작업을 한곳에서 고침
맡긴 일의 확인	진행 단계·승인·되묻기가 메시지에 섞이고, 알림 대상이 없는 예약 작업의 실패를 늦게 발견	채팅이 단계·승인·되묻기를 따로 표시하고, 받은편지함과 대시보드가 실패·멈춤 작업을 건수로 표시
조직 내 공유	스킬이 한 사람의 에이전트에 머물고, 호출 관계는 파일을 열어야 보임	스킬 허브 게시·설치, 스킬 연결도, 오피스·칸반 협업 공간

표의 세 행은 서로 독립된 문제처럼 보이지만 원인은 하나입니다. 즉 에이전트와 일하는 데 필요한 것이 대화만이 아닌데 창구가 대화만 담고 있다는 점입니다. 메신저를 다른 제품으로 바꿔도 세 공백은 그대로 남으므로, 대응 방법은 메신저 교체가 아니라 대화·구성·운영 상태를 한 화면에 모으는 별도 창구입니다.

조직이 직접 고칠 수 있는 것은 창구

시험 도입이 실제 업무 정착으로 이어지지 않는 원인은 여러 가지입니다. 모델의 성능, 데이터 준비, 예산처럼 조직이 혼자 바꾸기 어려운 원인도 있습니다. 그러나 그 가운데 조직이 직접 손댈 수 있는 것은 에이전트를 만들고, 일을 시키고, 결과를 지켜보는 업무 창구입니다. 창구가 대화만 담고 있으면 담당자는 에이전트의 구성을 볼 수 없고, 사람이 보지 않는 동안 실행된 정기 업무의 결과도 늦게 알게 됩니다. 반대로 창구가 구성과 실행 기록과 공유를 함께 담으면, 같은 모델과 같은 플랫폼을 쓰더라도 담당자가 에이전트를 넘겨받아 계속 쓸 수 있습니다. OPENMARU AAS가 새 모델이나 새 실행 엔진이 아니라 업무 담당자용 Studio라는 창구로 정의된 이유가 여기에 있습니다. 도입을 검토하는 조직은 어떤 모델을 쓸지와 함께, 담당자가 어떤 화면에서 에이전트를 만들고 시키고 지켜볼지를 같은 비중으로 따져 보는 것이 좋습니다.

7.1.2 콘솔 하나에서 프롬프트·스킬 제작·업무 지시를 마쳐야 한다는 초기 요구

제품 출발점이 된 세 가지 요구

OPENMARU AAS의 출발점이 된 요구는 콘솔 하나에 들어가면 프롬프트 입력, 스킬 제작, 에이전트에 대한 업무 지시를 모두 그 안에서 마칠 수 있어야 한다는 것이었습니다. 이 요구가 충족되면 에이전트와 일하려고 메신저를 따로 둘 필요도 줄어듭니다. 요구를 항목으로 나누면 다음 세 가지입니다.

요구	내용	현재 대응 기능
프롬프트 지시	대화로 일을 시키고, 진행 단계·승인·되묻기를 같은 화면에서 처리	채팅, 어느 화면에서나 여는 대화 창
스킬 제작	일하는 방법을 글로 정의하고, 고치고, 이전 버전으로 되돌림	스킬 만들기, 글 고치기와 이전 버전 40개 보관
업무 지시	즉시 대화, 칸반 카드, 정해진 시각의 예약 작업으로 일을 맡김	채팅, 칸반, 예약 작업 만들기

세 요구 가운데 스킬 제작은 화면에서 글을 고치는 범위를 넘어설 때가 있습니다. 예를 들어 규칙 점검 코드와 시험 코드가 필요한 스킬은 코딩 에이전트(예: Claude Code)가 만들고, 업무 담당자는 규칙을 문장으로 정하고 결과를 확인합니다. OPENMARU AAS는 「Claude Code 연결」 기능으로 사용자 PC의 Claude Code를 작업 공간에 연결해, 만든 스킬을 올리고 실제 AI 에이전트로 시험하게 합니다[S12]. 화면에서 직접 고치는 대상은 성격(SOUL), 설정, 스킬 문서, 예약 작업 지시문 같은 글이며, 저장 전에 바뀐 줄을 먼저 보여 주고 글마다 이전 버전을 40개까지 남깁니다[S14].

요구에서 나온 세 가지 설계 원칙

콘솔 하나에서 모든 일을 마치려면 콘솔이 에이전트 플랫폼을 깊이 다뤄야 합니다. 다만 그 과정에서 플랫폼을 고치면 이미 실행 중인 에이전트와 예약 작업에 영향을 주므로, OPENMARU AAS는 처음부터 세 가지 원칙을 정했습니다. 에이전트 플랫폼의 플러그인·설정·버전을 그대로 두고, 설정을 변경하는 경로는 플랫폼의 공식 API 하나로 제한하며, 비밀값은 어느 화면·로그·변경 기록에도 남기지 않는다는 원칙입니다. 이 원칙에 따라 메신저 연결 화면은 저장한 토큰을 다시 보여 주지 않고, 변경 기록은 누가 언제 무엇을 바꿨는지만 남기고 값은 기록하지 않습니다[S14].

이 원칙 때문에 OPENMARU AAS는 에이전트를 실행하는 주체가 아닙니다. 에이전트는 Hermes Agent 위에서 실행되고, OPENMARU AAS는 그 위에서 업무 담당자가 쓰는 화면과 권한을 맡습니다. 그래서 도입 조직은 기존 에이전트 플랫폼의 운영 방식을 바꾸지 않고 업무 담당자용 창구만 추가하는 형태로 시작할 수 있습니다.

7.2 만들기·시키기·지켜보기·나누기를 한 화면에 담은 업무 담당자용 Studio의 정의

OPENMARU AAS를 한 문장으로 정의하면, AI Agent 플랫폼에 연결해 업무 담당자가 자기 에이전트를 만들고, 일을 시키고, 지켜보고, 조직 안에서 나누는 오픈소스 Studio입니다. 네 동작은 각각 대응하는 화면이 있으며, 사용자 매뉴얼의 1부부터 7부까지가 이 동작 순서를 따라 구성돼 있습니다[S12]. 관리자 매뉴얼은 같은 제품을 구성·사용자·작업공간·모델·스킬 허브·점검의 관점에서 다시 설명합니다[S13].

동작	업무 담당자가 하는 일	대응 화면
만든다	에이전트의 성격·모델을 정하고, 스킬을 만들거나 허브에서 설치하고, 예약 작업을 정함	프로파일, 스킬 만들기·허브 설치, 예약 작업
시킨다	대화로 일을 맡기고, 승인 요청과 되묻기에 답하고, 파일을 주고받음	채팅, 받은편지함, 파일 서랍과 문서 뷰어

동작	업무 담당자가 하는 일	대응 화면
지켜본다	실패·멈춤 작업, 스킬의 호출 관계, 사용량, 변경 이력을 확인	대시보드, 스킬 연결도, 사용량, 변경 기록
나눈다	만든 스킬을 사내에 게시하고, 사람과 여러 에이전트가 한 과제를 나눠 맡음	스킬 허브 게시, 오피스·칸반·TFT 방

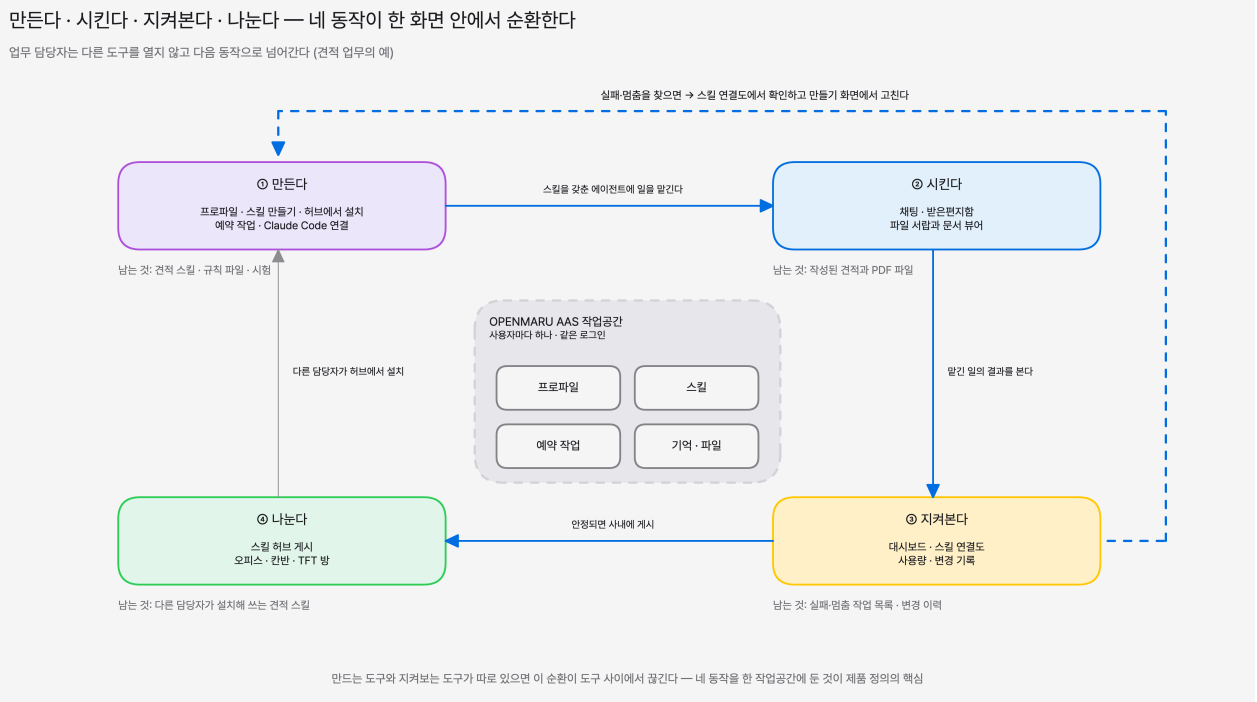


그림. 만든다·시킨다·지켜본다·나눈다 네 동작과 OPENMARU AAS의 대응 화면 — 한 작업공간 안에서 이어지는 순환

그림은 네 동작이 하나의 순환으로 이어진다는 점을 보여 줍니다. 지켜보기에서 발견한 문제가 다시 만들기 화면의 수정으로 연결되고, 수정이 안정되면 나누기로 넘어갑니다. 시장에서는 만드는 도구와 지켜보는 도구가 다른 제품으로 나뉘어 있는 경우가 많은데, OPENMARU AAS는 네 동작을 같은 로그인과 같은 작업공간 안에 둔 것이 정의의 핵심입니다.

7.2.1 사용자마다 하나씩 생기는 AI 에이전트 작업공간

사용자별 작업공간의 구성 요소

OPENMARU AAS에서는 사용자마다 AI 에이전트 작업공간이 하나씩 만들어지고, 여기에 설정·스킬·기억·파일이 저장됩니다[S12]. 작업공간은 Hermes Agent 설치 하나에 해당하며 프로파일을 하나 이상 가집니다. 프로파일마다 성격, 모델, 스킬, 도구, 예약 작업이 따로 있고, 실행 중인 프로파일이 채팅의 대화 상대이자 칸반 카드의 담당자인 AI 에이전트입니다[S14].

구성 요소	담기는 것	업무에서의 의미
프로파일	성격(SOUL.md)·모델·스킬·도구·MCP 서버·비밀값	업무별로 역할이 다른 에이전트를 나눠 둠

구성 요소	담기는 것	업무에서의 의미
스킬	일하는 방법을 적은 문서와 필요한 스크립트	업무 규칙이 사람이 아니라 작업공간에 남음
예약 작업	실행 주기, 실행할 스킬이나 지시문, 결과 전달 대상	정기 업무를 사람의 조작 없이 실행
기억과 파일	대화에서 쌓인 기억, 올린 파일과 에이전트가 만든 파일	같은 작업공간의 프로파일이 작업 폴더 하나를 공유

작업공간 단위 운영이 주는 이점

작업공간이 사용자 단위로 나뉘어 있으므로 한 사람의 설정 실수가 다른 사람의 에이전트에 번지지 않습니다. 관리자는 작업공간의 개수 한도와 상태를 관리하고 필요하면 재시작하며[S13], 담당자가 바뀔 때 에이전트와 스킬을 조직 안에서 넘길 수 있습니다. 다만 공개본 Docker Compose 구성은 작업공간 하나를 실행하고, 사용자마다 작업공간을 새로 만드는 기능은 쿠버네티스 배포에서만 제공합니다[S14]. 따라서 여러 부서가 함께 쓰는 시범 도입이라면 처음부터 쿠버네티스 배포를 전제로 설치 계획을 세우는 편이 맞습니다.

7.2.2 만들기·시키기·지켜보기·나누기 네 동작이 이어지는 업무 담당자의 하루

견적 업무로 본 네 동작의 순서

네 동작이 실제 업무에서 어떤 순서로 이어지는지는 6장의 견적 작성 사례로 확인할 수 있습니다. 영업 업무 담당자는 견적 규칙을 문장으로 정하고, 연결한 코딩 에이전트가 그 문장을 규칙 파일·점검 코드·시험으로 만듭니다. 그 뒤 업무 담당자는 채팅에서 견적을 요청하고, 에이전트가 되묻는 항목에 답하고, 최종 확인 단계에서 승인합니다. 이 흐름을 네 단계로 정리하면 다음과 같습니다.

단계	동작	쓰는 화면	남는 결과
1	만든다	프로파일, 스킬, Claude Code 연결	견적 스킬과 규칙 파일, 시험
2	시킨다	채팅, 받은편지함	작성된 견적과 PDF 파일
3	지켜본다	대시보드, 예약 작업, 변경 기록	실패·멈춤 작업 목록과 변경 이력
4	나눈다	스킬 허브 게시, 칸반	다른 담당자가 설치해 쓰는 견적 스킬

단계 사이를 잇는 화면 연결

네 단계가 한 화면에 있다는 것은 단계 사이를 이동할 때 다른 도구를 열 필요가 없다는 뜻입니다. 예를 들어 지켜보기 단계에서 실패한 예약 작업을 누르면 해당 예약 작업 화면으로 바로 이동하고[S14], 그 작업이 부르는 스킬을 스킬 연결도에서 확인한 뒤 같은 화면의 코드 탭에서 고칠 수 있습니다. 수정한 스킬이 안정되면 스킬 허브에 게시해 다른 부서가 설치하게 합니다. 이 구조에서는 업무 담당자 한 사람이 만들기부터 나누기까지 한 업무를 끝까지 책임지는 구조가 됩니다.

7.3 에이전트 플랫폼 대시보드와 OPENMARU AAS의 역할 구분 — 전체 관리와 사용자별 에이전트 운영

Hermes Agent는 자체 대시보드를 제공하고, 이 대시보드는 플랫폼 설치 전체의 프로파일·예약 작업·스킬·키·로그·사용량을 관리합니다. OPENMARU AAS도 같은 정보를 다루므로 두 화면이 겹친다고 볼 수 있지만, 쓰는 사람과 관리 범위가 다릅니다. 플랫폼 대시보드는 설치 전체를 관리하는 운영자용 화면이고, OPENMARU AAS는 사용자마다 자기 에이전트를 운영하는 업무 담당자용 화면입니다. 그래서 IT 담당자는 기존 플랫폼 운영을 그대로 두고 업무 담당자에게 OPENMARU AAS를 열어 주는 방식으로 역할을 나눌 수 있습니다.

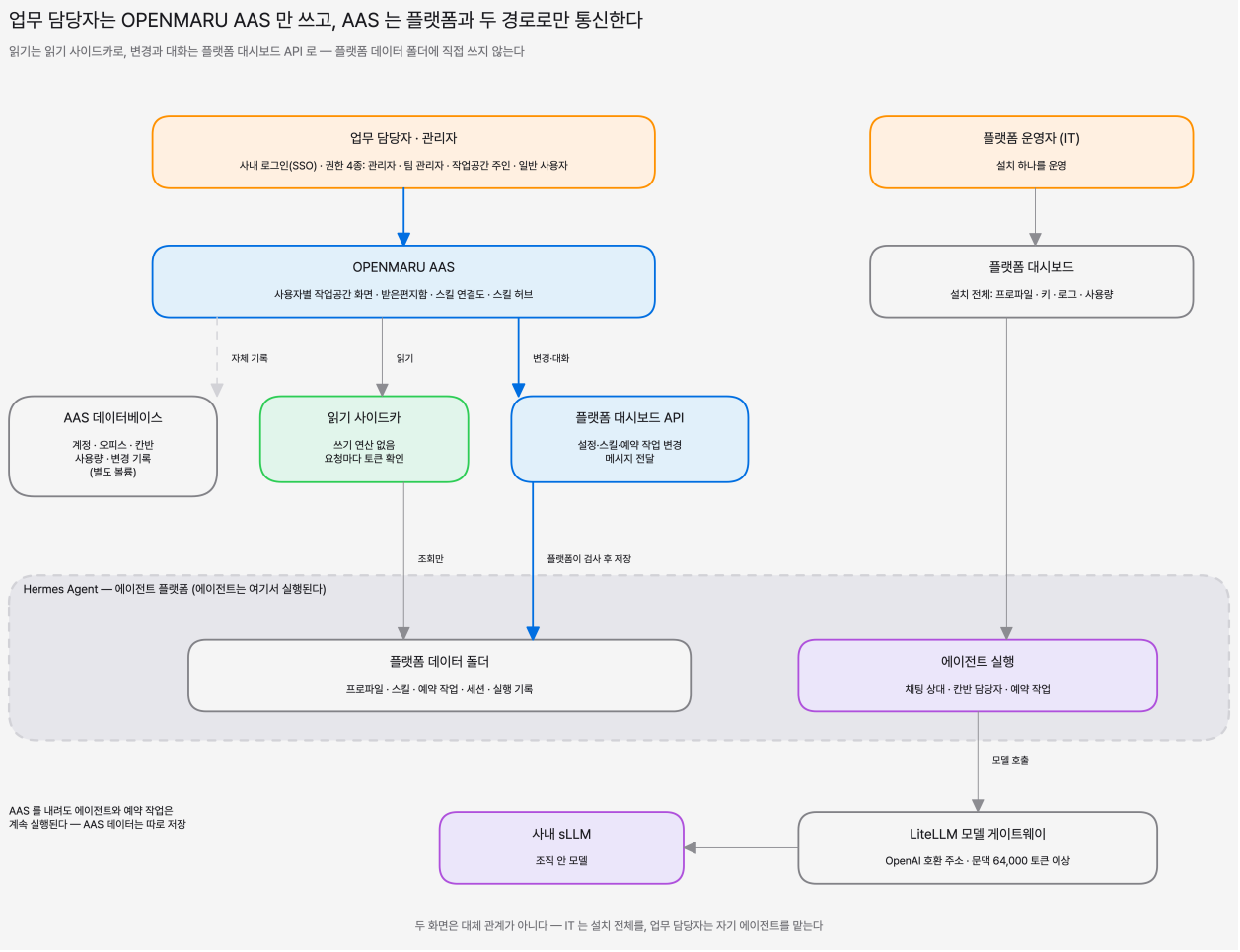


그림. 에이전트 플랫폼(Hermes Agent)–OPENMARU AAS–업무 담당자 구성도 — 읽기는 읽기 사이드카, 변경과 대화는 플랫폼 대시보드 API를 거친다

구성도는 업무 담당자가 OPENMARU AAS 화면만 쓰고, OPENMARU AAS가 플랫폼과 두 경로로 통신하는 구조를 보여 줍니다. 모델 호출은 이 두 경로와 별개로 에이전트가 LiteLLM 모델 게이트웨이를 거쳐 사내 모델을 부르는 방식으로 이뤄집니다[S13].

7.3.1 플랫폼 데이터 폴더에 직접 쓰지 않고 공식 API로만 바꾸는 설치 구조

읽기 경로와 변경 경로의 분리

OPENMARU AAS는 읽기와 바꾸기를 서로 다른 경로로 처리합니다. 실행 기록, 세션, 스킬, 예약 작업, 사용량을 읽을 때는 Hermes Agent 옆에서 실행되는 읽기 사이드카에 묻고, 이 사이드카에는 쓰기 연산이 없습니다.

반면 설정을 바꾸거나 에이전트와 대화할 때는 Hermes 대시보드 API를 부르고, Hermes Agent가 변경을 검사해 저장합니다. OPENMARU AAS는 Hermes 데이터 폴더에 직접 쓰지 않습니다[S14].

경로	거치는 구성 요소	하는 일	플랫폼 데이터 변경
읽기	읽기 사이드카(요청마다 토큰 확인)	실행 기록·세션·스킬·예약 작업·사용량 조회	없음
변경·대화	Hermes 대시보드 API	설정·스킬·예약 작업 변경 요청, 메시지 전달	Hermes Agent가 검사 후 저장
자체 데이터	OPENMARU AAS 데이터베이스(별도 볼륨)	계정·오피스·칸반·사용량·변경 기록 보관	없음

기존 플랫폼 운영과의 충돌 여부

이 구조에서 OPENMARU AAS가 플랫폼에 남기는 변경은 모두 플랫폼 자신의 검사를 통과한 공식 API 호출입니다. 따라서 플랫폼을 업그레이드하거나 설정 형식이 바뀌어도 OPENMARU AAS가 데이터 파일을 직접 고쳐 손상시키는 경우는 생기지 않습니다. OPENMARU AAS의 계정, 오피스, 칸반, 변경 기록은 별도 볼륨에 따로 저장되므로 OPENMARU AAS를 내려도 Hermes Agent의 에이전트와 예약 작업은 계속 실행됩니다[S14]. 이때 IT 담당자가 확인할 조건은 하나입니다. Hermes Agent가 도구를 쓰려면 문맥이 64,000 토큰 이상인 OpenAI 호환 LLM 주소가 필요하므로, 사내 모델이 이 조건을 만족하는지 먼저 확인해야 합니다[S14].

7.3.2 전체 관리 화면과 사용자별 에이전트 운영 화면의 기능 대비

두 화면의 사용자와 관리 범위

플랫폼 대시보드와 OPENMARU AAS는 일부 기능이 겹치지만, 누가 쓰고 무엇을 범위로 삼는지가 다릅니다. 플랫폼 대시보드는 설치 하나를 운영하는 사람이 프로파일·키·로그·사용량을 관리하는 화면입니다. 반면 OPENMARU AAS는 사내 로그인(SSO)으로 들어온 여러 사용자가 각자 자기 작업공간을 보고, 관리자·팀 관리자·작업공간 주인·일반 사용자 네 가지 권한으로 볼 수 있는 범위가 정해집니다[S13].

항목	플랫폼 대시보드	OPENMARU AAS
주 사용자	플랫폼 운영자	업무 담당자, 관리자
관리 범위	설치 전체	사용자별 작업공간
로그인과 권한	운영자 계정	SSO와 권한 4종
주요 화면	프로파일·예약 작업·스킬·키·로그·사용량	같은 정보에 더해 받은편지함, 실패·멈춤 모아 보기, 스킬 연결도, 협업 공간, 스킬 허브
변경 방식	플랫폼 직접 변경	플랫폼 공식 API 경유

겹치는 기능을 대하는 도입 판단

한 사람이 에이전트 플랫폼 하나를 운영한다면 플랫폼 대시보드만으로 충분할 수 있습니다. 하지만 여러 업무 담당자에게 에이전트를 나눠 주고, 각자 무엇이 실패했는지와 스킬이 무엇을 부르는지를 먼저 보여 줘야 한다면 사

용자별 화면이 필요해집니다. 두 화면은 대체 관계가 아니라 층이 다른 도구로 보는 편이 정확하며, IT 담당자는 플랫폼 대시보드로 설치 전체를 관리하고 업무 담당자는 OPENMARU AAS로 자기 에이전트를 운영하는 분담이 기본 형태입니다. 이 분담이 맞는 조직인지는 업무 담당자 여러 명이 각자 에이전트를 맡는 계획이 있는지로 판단할 수 있습니다.

8장: AI Agent 개발 도구·빌더와 비교한 특징과 차별점

7장의 제품 정의를 받아들이더라도, 도입을 검토하는 조직은 이미 시장에 나온 다른 에이전트 도구와 견줘 봐야 합니다. AI Agent를 만드는 도구는 이미 많고, 대형 소프트웨어 공급사도 자사 제품 안에 에이전트 제작 기능을 넣고 있습니다. 2장에서 본 것처럼 Gartner는 2026년 말까지 기업용 애플리케이션의 40%가 업무 특화 AI 에이전트를 통합할 것으로 전망했고(현재 5% 미만)[S7], 2025년 AI 하이프 사이클에서는 AI 에이전트를 기대의 정점에 놓았습니다[S9]. 선택지가 많고 기대가 높은 시기에는 도구 이름보다 도구가 어떤 일을 돕는지, 어디에 설치되는지로 비교 기준을 먼저 정해야 합니다. 에이전트 도구를 여섯 유형으로 나누고 캔버스 조립 방식과 말로 정의하는 방식을 견주면, OPENMARU AAS가 앞서는 영역과 아직 채우지 못한 영역을 같은 비중으로 볼 수 있습니다. 경쟁 제품은 실명 대신 유형으로 부르며, 비교 내용은 각 유형의 공개 자료 기재를 기준으로 합니다.

8.1 만들기 중심·지켜보기 중심과 SaaS·조직 안 설치로 나눈 에이전트 도구 여섯 유형

에이전트 도구를 비교할 때 쓸 수 있는 기준은 두 가지입니다. 첫 번째 기준은 도구가 주로 돕는 일이 에이전트를 만드는 것인지, 이미 실행 중인 에이전트를 지켜보고 관리하는 것인지입니다. 두 번째 기준은 도구를 공급사의 클라우드 서비스(SaaS)로 쓰는지, 조직 안에 설치하는지입니다. 시장에서 만드는 도구에는 주로 Studio와 Builder라는 이름이 붙고, 지켜보는 도구에는 Console과 Control이라는 이름이 붙어 두 기능이 다른 제품으로 나뉘어 있는 경우가 많습니다. 두 기준을 가로세로로 놓으면 지금 시장의 에이전트 도구는 여섯 유형으로 나눌 수 있습니다.

8.1.1 유형별 주된 사용자와 대표 사용 장면

만들기 중심의 세 유형

SaaS 업무용 에이전트 빌더는 현업 부서 담당자가 원하는 일을 문장으로 적으면 에이전트가 만들어지는 방식을 앞세웁니다. 공급사의 협업 도구와 클라우드 안에서 실행되고, 에이전트 스토어·관리자 통제·지출 한도 같은 조직 기능이 완성도 높게 갖춰져 있습니다. 조직 안 설치형 상용 AI 플랫폼은 망분리 조직이 생성형 AI 서비스, 에이전트 빌더, 사내 스토어를 한 공급사에서 함께 받는 형태입니다. 국내 시장에서는 이 유형이 조직 안 설치를 기본 선택지로 제공합니다. 캔버스 조립형 빌더는 노드를 선으로 이어 RAG 챗봇이나 업무 자동화 흐름을 조립하는 오픈소스 도구로, 개발자나 자동화 담당자가 자체 서버에 설치해 씁니다.

시키기·지켜보기 중심의 세 유형

채팅 워크스페이스는 LLM 앞단에서 전 직원이 쓰는 대화 화면이며, 그룹 권한과 채널 기능이 잘 갖춰져 있지만 에이전트 플랫폼의 구성과 운영은 다루지 않습니다. 에이전트 관제·거버넌스 도구는 IT·보안팀이 조직의 에이전트를 등록하고, 호출을 추적하고, 감사 기록을 남기는 데 씁니다. 이 유형은 실행 상태를 깊이 보여 주지만 업무 담당자가 에이전트를 만드는 화면은 제공하지 않습니다. 에이전트 플랫폼 전용 콘솔은 특정 오픈소스 에이전트 런타임의 프로파일·예약 작업·스킬을 다루는 화면으로, 대부분 그 플랫폼을 운영하는 한 사람을 전제로 설계됐습니다. 그 밖에 코드로 에이전트를 짜는 개발 프레임워크도 있지만, 화면이 아니라 개발자가 쓰는 라이브러리이므로 이 지도에는 넣지 않았습니다.

여섯 유형의 주된 사용자와 대표 사용 장면을 한 표로 정리하면 다음과 같습니다.

유형	주된 사용자	대표 사용 장면	주로 돕는 일	설치 형태
SaaS 업무용 에이전트 빌더	현업 부서 담당자	문장으로 에이전트를 만들어 협업 도구 안에서 사용	만들기	SaaS 중심
조직 안 설치형 상용 AI 플랫폼	IT 부서와 현업	생성형 AI 서비스·빌더·스토어를 한 공급사에서 도입	만들기	조직 안
캔버스 조립형 빌더	개발자·자동화 담당	노드를 이어 RAG 챗봇·자동화 흐름 조립	만들기	자체 호스팅
채팅 워크스페이스	전 직원	모델 앞단의 대화 화면과 그룹 권한	시키기	자체 호스팅
에이전트 관제·거버넌스 도구	IT·보안팀	에이전트 등록·호출 추적·감사	지켜보기	SaaS와 조직 안
에이전트 플랫폼 전용 콘솔	플랫폼 운영자 한 사람	한 런타임의 프로파일·예약 작업·스킬 관리	지켜보기·구성	조직 안

표에서 예약 실행, 사내 공유, 여러 에이전트의 협업은 어느 한 유형의 특징으로 적지 않았습니다. 상용과 오픈소스를 가리지 않고 많은 도구가 이미 제공하는 기본 기능이기 때문입니다. 따라서 이 세 기능은 도구를 고르는 기준이라기보다 갖추지 않았을 때 제외하는 최소 조건으로 보는 편이 맞습니다.

8.1.2 여섯 유형 지도 위에 놓인 OPENMARU AAS의 위치

두 축에서 본 OPENMARU AAS의 위치

OPENMARU AAS는 에이전트 플랫폼 전용 콘솔에서 출발했지만, 겨냥하는 사용자는 SaaS 업무용 에이전트 빌더와 같은 업무 담당자입니다. 같은 오픈소스 플랫폼 위에 설치된다는 점은 전용 콘솔과 같고, 사용자마다 작업공간을 나누고 SSO와 권한 4종으로 접근 범위를 정한다는 점은 다릅니다[S13]. 만들기와 지켜보기 축에서는 한쪽 끝에 서지 않고 두 기능을 한 화면에 둡니다. 설치 축에서는 조직 안 설치이며, 모델 호출은 사내 모델 게이트웨이를 거칩니다[S13].

축	OPENMARU AAS의 위치	같은 위치에 있는 유형
만들기 ↔ 지켜보기	두 기능을 한 화면과 한 작업공간에 둠	해당 유형 없음(대부분 한쪽 중심)
SaaS ↔ 조직 안	조직 안 설치, 사내 sLLM 연결	조직 안 설치형 상용 AI 플랫폼, 플랫폼 전용 콘솔
플랫폼 관계	쓰던 오픈소스 에이전트 플랫폼 위에 설치	플랫폼 전용 콘솔
대상 사용자	업무 담당자와 관리자	SaaS 업무용 에이전트 빌더

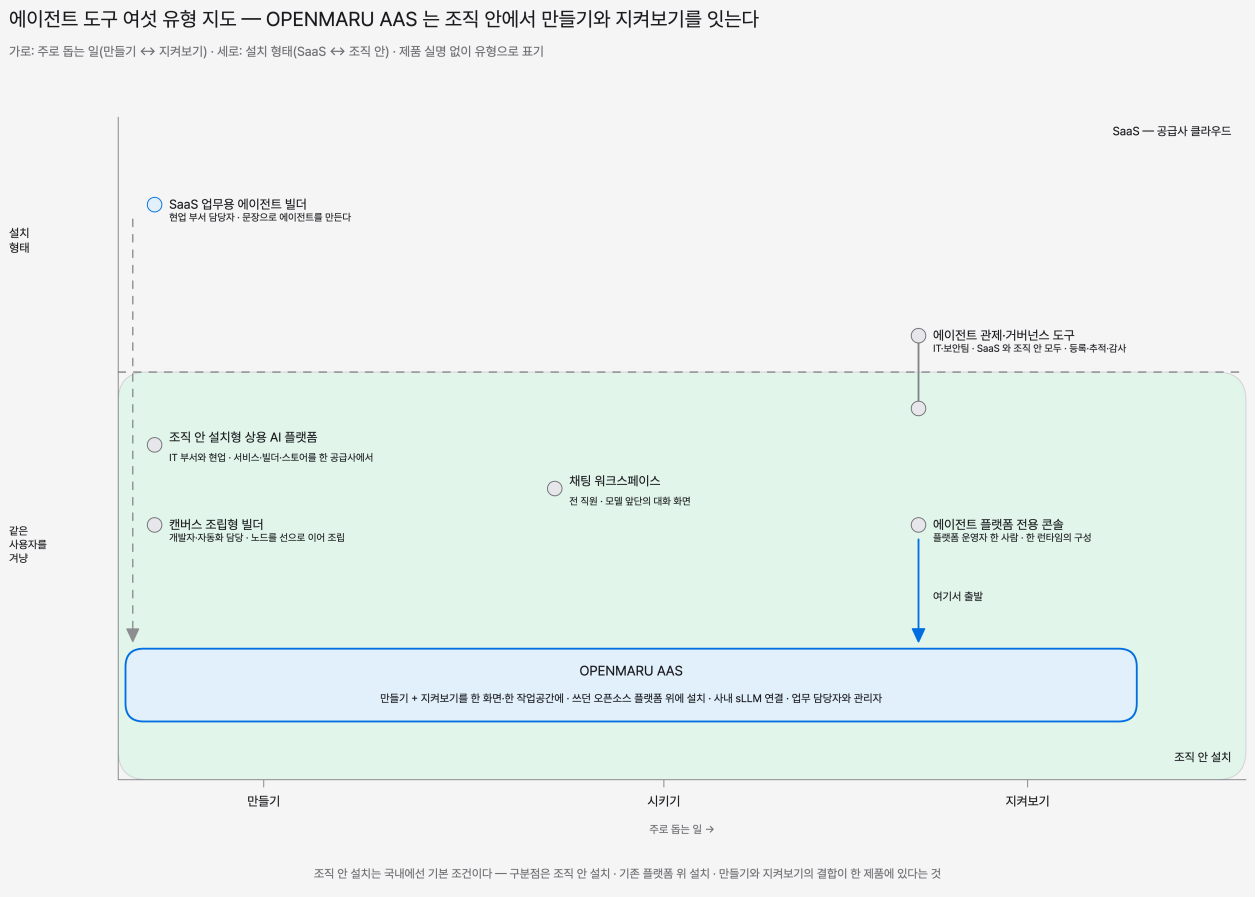


그림. 만들기↔지켜보기, SaaS↔조직 안 두 축 위의 에이전트 도구 여섯 유형과 OPENMARU AAS의 위치 (실명 없음)

지도에서 확인할 점은 조직 안 설치 그 자체가 OPENMARU AAS만의 특징이 아니라는 것입니다. 국내 시장에서는 조직 안 설치형 상용 AI 플랫폼이 이미 같은 축에 있으므로, 조직 안 설치는 국내 경쟁에서 차별점이라기보다 기본 조건에 가깝습니다. OPENMARU AAS가 지도에서 다른 유형과 구분되는 이유는 조직 안 설치, 기존 오픈소스 플랫폼 위의 설치, 만들기와 지켜보기의 결합이 한 제품에 함께 있다는 데 있습니다.

8.2 「빌더는 이미 많다」는 반론과 캔버스 조립·말로 정의하는 제작 방식의 차이

도입을 검토하는 조직이 가장 먼저 하는 질문은 에이전트 빌더가 이미 많은데 왜 또 하나가 필요하냐는 것입니다. 이 질문에는 기능 목록보다 누가 에이전트를 만들고 그 결과가 어떤 형태로 남는지로 답하는 편이 정확합니다. Anthropic은 미리 정한 코드 경로로 LLM과 도구를 조율하는 시스템을 워크플로, LLM이 스스로 과정과 도구 사용을 정하는 시스템을 에이전트로 구분합니다[S11]. 캔버스 조립 방식은 주로 워크플로를 사람이 그림으로 설계하는 방식이고, 말로 정의하는 방식은 업무 담당자의 문장에서 출발해 에이전트가 쓸 규칙과 코드를 만드는 방식입니다.

8.2.1 노드를 연결하는 캔버스 빌더의 제작 주체와 변경 비용

캔버스 조립 방식의 강점

캔버스 빌더는 처리 흐름이 그림으로 보인다는 강점이 있습니다. 입력을 받는 노드, 문서를 검색하는 노드, LLM을 부르는 노드, 결과를 보내는 노드를 선으로 이으면 흐름 전체를 한 화면에서 확인할 수 있고, 절차가 고정된

업무라면 빠르게 실행 가능한 결과물을 얻습니다. 미리 정한 경로를 따르는 워크플로는 결과를 예측하기 쉽고, Anthropic도 가능한 한 단순한 해법에서 시작하고 필요할 때만 복잡도를 높이라고 권합니다[S11]. 따라서 사내 규정 검색 챗봇처럼 흐름이 일정한 업무에는 캔버스 조립이 여전히 좋은 선택입니다.

흐름 설계 능력이 필요한 제작 주체

다만 캔버스 조립은 결국 흐름을 설계할 수 있는 사람을 요구합니다. 분기 조건, 변수 전달, 오류 처리, 재시도를 노드와 선으로 표현해야 하므로 업무 규칙을 아는 사람과 흐름을 그리는 사람이 다를 때가 많습니다. 업무 규칙이 바뀌면 어느 노드의 프롬프트와 조건을 고쳐야 하는지 찾아야 하고, 규칙이 여러 노드에 나뉘어 적혀 있으면 수정 뒤 전체 흐름을 사람이 다시 시험해야 합니다. 결과물은 그 빌더의 실행 엔진 안에서만 동작하는 흐름 정의 파일이므로, 도구를 바꾸면 다시 조립해야 합니다. 또한 인기 있는 캔버스 빌더 가운데 일부는 OSI(Open Source Initiative) 승인 라이선스가 아니어서, 제품에 넣거나 납품할 때 라이선스 조건을 따로 확인해야 합니다.

8.2.2 업무 담당자 문장에서 출발해 규칙 파일과 시험으로 남는 스킬 제작

문장에서 규칙 파일·코드·시험으로

OPENMARU AAS에서 스킬을 만드는 출발점은 업무 담당자의 문장입니다. 6장의 견적 작성 사례에서 영업 업무 담당자는 「공공기관 견적은 VAT 포함」, 「조달 품목 단가는 등록가 ÷ 1.1」 같은 규칙을 말로 정했고, 연결한 코딩 에이전트가 그 문장을 규칙 파일, 점검 코드, 시험, 사람이 읽는 규칙 페이지로 만들었습니다. 규칙을 바꿀 때도 업무 담당자는 문장만 고치고, 코딩 에이전트가 규칙 파일과 시험을 함께 고칩니다. 결과물은 Hermes Agent의 스킬 형식인 문서와 스크립트로 남으므로 특정 빌더의 실행 엔진에 종속되지 않고, 화면에서 고친 글은 이전 버전을 40개까지 남겨 비교하거나 되돌릴 수 있습니다[S14].

만드는 단계와 일하는 단계의 구분

이 방식은 만드는 단계와 일하는 단계를 나눠 운영합니다. 만드는 단계에서는 코딩 에이전트가 규칙과 예시 자료만 다루며, 현재는 외부 코딩 에이전트인 Claude Code를 쓰며, OPENMARU AAS는 「Claude Code 연결」 기능으로 그 Claude Code가 만든 스킬을 작업공간에 올려 실제 AI 에이전트로 시험하게 합니다[S12]. 폐쇄망 조직이라면 코딩 에이전트를 사내 코딩 모델로 바꿀 수 있습니다. 일하는 단계에서는 사내 sLLM 위의 에이전트가 완성된 스킬로 매일 업무를 처리하므로 고객·금액 자료가 조직 밖 LLM으로 나가지 않습니다. 하지만 이 방식에도 부담이 있습니다. 코딩 에이전트가 만든 코드가 규칙을 제대로 반영했는지는 시험 결과로 확인해야 하고, 시험이 무엇을 검사하는지 읽을 수 있는 사람이 조직에 한 명은 있어야 합니다.

두 제작 방식을 같은 기준으로 비교하면 다음과 같습니다.

기준	캔버스 조립 방식	말로 정의하는 방식(OPENMARU AAS)
만드는 사람	흐름을 설계할 수 있는 개발자·자동화 담당	업무 담당자(문장) + 코딩 에이전트(코드)
출발점	노드와 선으로 그린 처리 흐름	업무 규칙을 적은 문장
변경 방법	해당 노드의 프롬프트·조건을 찾아 수정	문장을 고치면 규칙 파일과 시험이 함께 바뀜

기준	캔버스 조립 방식	말로 정의하는 방식(OPENMARU AAS)
결과물 형태	빌더 실행 엔진 안의 흐름 정의	규칙 파일·스크립트·시험이 있는 스킬
검증 방법	수정 뒤 사람이 흐름을 다시 실행해 확인	시험 코드로 확인, 새 규칙을 지난 처리 결과에 다시 적용
잘 맞는 업무	절차가 고정된 검색·자동화	판정 규칙이 분명하고 자주 바뀌는 업무
남는 부담	흐름 설계 인력, 도구 종속	코딩 에이전트 의존, 시험을 읽을 사람

표의 마지막 두 행이 선택 조건입니다. 절차가 고정돼 있고 흐름 설계 인력이 있다면 캔버스 조립이 빠르고, 판정 규칙이 업무 담당자의 머릿속에 있고 자주 바뀐다면 문장에서 출발하는 방식이 변경 비용을 줄입니다.

8.3 OPENMARU AAS가 앞서는 영역 — 조직 안 설치·사용자별 작업공간·운영 가시성

OPENMARU AAS의 특징은 하나의 기능보다 기능의 조합에 있습니다. 조직 안에 설치되고, 쓰던 오픈소스 에이전트 플랫폼을 고치지 않고 그 위에 설치되며, 업무 담당자가 만드는 화면과 그 에이전트가 무엇을 하다 실패했는지 지켜보는 화면이 하나라는 점입니다. 다음 표는 앞서는 영역 다섯 행과 아직 채우지 못한 영역 다섯 행을 같은 표에 놓은 것이며, 부족한 영역은 8.4절에서 같은 비중으로 설명합니다.

비교 기준	OPENMARU AAS 현재	판정	이 기준에서 앞선 유형
조직 안 설치와 사내 sLLM 연결	OpenAI 호환 사내 모델 주소 하나로 연결, 모델 게이트웨이 경유	앞섬(SaaS 대비)	조직 안 설치형 상용 AI 플랫폼과 동등
쓰던 에이전트 플랫폼과의 관계	플랫폼을 고치지 않고 공식 API로만 변경	앞섬	해당 없음(상용 유형은 자사 런타임 일체형)
사용자별 작업공간	사용자마다 작업공간과 프로파일	앞섬	해당 없음(대부분 조직 단위 격리)
만들기와 지켜보기의 결합	같은 화면에서 스킬 수정과 실패·승인·사용량 확인	앞섬	해당 없음(대부분 한쪽 중심)
예약 작업 상태와 스킬 호출 관계 표시	실패·멈춤 건수, 스킬 연결도	앞섬	관제 도구가 일부 제공
연결되는 에이전트 플랫폼	Hermes Agent 하나	부족	여러 런타임을 지원하는 오픈소스 콘솔
공개 적용 사례	많지 않음	부족	상용 유형 전부
사내 마켓	스킬 허브와 게시 검토, 결제 없음	부족	SaaS 빌더, 조직 안 설치형 상용 플랫폼

비교 기준	OPENMARU AAS 현재	판정	이 기준에서 앞선 유형
비용 통제	사용량 조회 중심	부족	지출 한도·부서별 할당을 제공하는 상용 유형
기능 폭과 완성도	기능별 깊이가 고르지 않음	부족	상용 유형 전부

표의 「앞섬」은 각 유형의 공개 자료에서 같은 기능을 전면에 내세운 사례를 찾지 못했다는 뜻이며, 모든 제품에 그 기능이 없다는 뜻은 아닙니다. 따라서 차별점으로 판단하려면 시범 도입 단계에서 실제 화면을 놓고 비교하는 절차가 필요합니다.

8.3.1 조직 안 sLLM과 폐쇄망 배포를 전제로 한 설치 구조

SaaS 유형과 달라지는 자료 경로

SaaS 업무용 에이전트 빌더는 공급사의 클라우드와 자사 런타임을 전제로 하므로, 외부 서비스를 쓸 수 없는 조직에서는 선택지가 되기 어렵습니다. OPENMARU AAS는 4장에서 설명한 조직 안 sLLM 구성을 그대로 전제로 합니다. LLM 연결은 사내 vLLM, LiteLLM, Ollama처럼 OpenAI 호환 주소 하나면 되고, 도구 사용을 위해 문맥이 64,000 토큰 이상이어야 합니다[S14]. 에이전트는 LiteLLM을 거쳐 AI 모델을 부르며, 대화용 모델 목록에는 운영자가 정한 이름 규칙에 맞는 게이트웨이 모델만 표시됩니다[S13].

폐쇄망 배포를 위한 장치

폐쇄망에서는 설치 뒤 외부 주소로 요청이 나가지 않는지가 보안성 검토의 첫 질문입니다. OPENMARU AAS는 3D 오피스에 쓰는 글꼴과 모델을 저장소 안에 넣고, 빌드할 때 외부 요청이 0건인지 검사합니다[S14]. 파일 서랍의 문서 미리 보기도 브라우저 안에서 그리며 파일을 외부 변환 서비스로 보내지 않습니다[S14]. 관리자 화면에는 질문 가드레일과 위험한 명령 차단이라는 두 가지 안전장치가 있고, 이 안전장치는 화면에서 끌 수 없습니다[S13]. 다만 조직 안 설치는 국내 상용 AI 플랫폼도 제공하므로, 그 유형과의 차이는 설치 위치보다 쓰던 오픈소스 플랫폼 위에 설치된다는 점과 코드가 Apache-2.0으로 공개돼 보안성 검토 때 직접 열어 볼 수 있다는 점에 있습니다.

8.3.2 만들기와 지켜보기를 한 화면에서 잇는 사용자별 작업공간

만든 에이전트의 운영 상태를 보여 주는 화면

시장에서 만드는 도구와 지켜보는 도구가 다른 제품으로 나뉘어 있으면, 업무 담당자는 에이전트를 만든 뒤 그 에이전트가 정해진 시각에 제대로 실행됐는지를 다른 팀의 관제 화면에 물어야 합니다. OPENMARU AAS에서는 만든 에이전트의 예약 작업, 승인 요청, 사용량을 만든 사람이 같은 화면에서 봅니다. 화면 위쪽 받은편지함에는 명령 승인 요청, 에이전트의 질문, 실패한 예약 작업이 건수로 표시되고, 승인과 답은 받은편지함에서 바로 할 수 있습니다[S14]. 대시보드는 작업공간의 상태와 30일 추이를 보여 주고, 사용량 화면은 세션과 토큰 사용량을, 변경 기록은 누가 언제 무엇을 바꿨는지를 값 없이 보여 줍니다[S12][S14].

스킬 호출 관계와 사용자별 분리

스킬 연결도는 스킬이 어디서 불리는지(예약 작업, 다른 스킬)와 무엇을 쓰는지(스크립트, 라이브러리, API 키·토큰, 외부 주소, MCP 서버)를 스킬 문서와 스크립트에서 읽어 그림으로 보여 줍니다[S14]. 설계 단계에서 그린 흐름이 아니라 실제 스킬 파일에서 뽑은 관계이므로, 담당자가 바뀐 뒤에도 그 스킬이 어떤 외부 시스템에 접속하는지 확인할 수 있습니다. 또한 조사한 상용 유형은 대부분 조직(테넌트·워크스페이스) 단위로 격리하고 그

안을 권한으로 나누는 반면, OPENMARU AAS는 사용자마다 작업공간을 따로 줍니다. 다만 조직 전체의 정책 통제와 감사 보고는 에이전트 관제·거버넌스 유형이 더 깊게 다루므로, 보안팀이 회사 전체 에이전트를 한곳에서 통제해야 하는 조직이라면 관제 도구를 함께 검토해야 합니다.

8.4 OPENMARU AAS가 아직 채우지 못한 영역 — 단일 플랫폼 연동·결제형 마켓 부재·조회 중심 비용 통제

OPENMARU AAS가 아직 맞지 않는 조직도 분명히 있습니다. 연결되는 에이전트 플랫폼이 하나이고, 공개 적용 사례가 많지 않으며, 사내 마켓에 결제가 없고, 비용 통제가 조회 중심이라는 점입니다. Gartner는 에이전틱 AI 프로젝트가 취소되는 원인으로 비용 상승과 부족한 위험 통제를 함께 꼽았으므로[S4], 이 한계는 도입 판단에서 앞서는 영역과 같은 무게로 봐야 합니다. 다음 표는 한계별 현재 상태와 보완 방향, 그리고 시범 도입 단계에서 쓸 수 있는 대안을 정리한 것입니다.

한계	현재 상태	보완 방향	시범 도입 단계의 대안
연결 플랫폼	Hermes Agent 하나	두 번째 에이전트 플랫폼 연결(로드맵 마지막 단계)	Hermes Agent를 쓰는 조직·업무로 범위를 정함
적용 사례	공개 사례가 많지 않음	판정 기준이 분명한 업무부터 사례 축적	업무 하나를 정해 결과를 직접 확인
사내 마켓	스킬 허브(공용·제한 구역, 게시 검토), 결제 없음	마켓 운영 단계에서 권한과 결제를 다룸	사내 공유 용도로만 사용
비용 통제	세션·토큰 사용량 조회	한도 기능은 아직 제공하지 않음	모델 게이트웨이의 키별 예산 기능과 함께 운영

8.4.1 연동 에이전트 플랫폼 하나와 아직 좁은 적용 사례 범위

Hermes Agent 하나로 한정된 연결 범위

OPENMARU AAS는 여러 에이전트 플랫폼에 연결하는 것을 목표로 설계했지만, 지금 연결되는 플랫폼은 오픈소스 Hermes Agent 하나입니다[S14]. 플랫폼을 다루는 코드는 일부 모듈에 모아 두었으나 다른 플랫폼용 연결은 아직 구현하지 않았습니다. 따라서 이미 다른 에이전트 런타임을 표준으로 정한 조직이라면 지금의 OPENMARU AAS는 맞지 않습니다. 반면 여러 런타임을 지원하는 오픈소스 콘솔은 이미 있으므로, 런타임을 여럿 함께 써야 하는 조직은 그 유형이 더 적합합니다.

공개 적용 사례와 배포 범위의 제약

공개된 적용 사례도 아직 많지 않습니다. 이 백서의 대표 사례인 견적 작성은 판정 규칙이 분명한 업무이고, 그 업무를 정의한 영업 업무 담당자는 기술 배경이 있습니다. 개발 경험이 없는 부서 담당자가 스킬을 직접 정의해 운영한 공개 사례는 아직 없으므로, 시범 도입에서는 판정 기준이 분명한 업무 하나를 골라 업무 담당자가 말로 정의한 규칙이 코드와 시험으로 옮겨지는지 직접 확인하는 순서를 권합니다. 또한 공개본 Docker Compose 구성은 작업공간 하나를 실행하고, 사용자마다 작업공간을 만들거나 스킬 허브에서 스킬 묶음을 설치하는 기능은 쿠버네티스 배포에서만 제공합니다[S14]. 이 범위 차이는 11장의 오픈소스 공개 범위와 함께 확인해야 합니다.

8.4.2 결제형 마켓 부재와 조회 중심 비용 통제의 보완 방향

결제 없이 운영되는 사내 스킬 허브

OPENMARU AAS의 스킬 허브는 공용 구역과 제한 구역을 나누고, 관리자가 게시 요청을 검토한 뒤 공개하는 절차를 갖추고 있습니다[S13]. 부서가 만든 스킬을 다른 부서가 설치해 쓰는 사내 공유에는 이 기능으로 충분합니다. 하지만 SaaS 업무용 에이전트 빌더와 조직 안 설치형 상용 플랫폼이 제공하는 에이전트 스토어처럼 결제와 정산이 붙은 마켓은 아직 없습니다. 부서 간 비용 정산이나 외부 판매까지 마켓에서 처리해야 하는 조직이라면 지금은 맞지 않으며, 로드맵은 협업 방의 생성·삭제 다음 단계에서 마켓 운영의 권한과 결제를 다루고 그다음에 두 번째 에이전트 플랫폼 연결로 이어집니다(12장).

사용량 조회 중심의 비용 관리

비용 통제는 현재 조회 중심입니다. 사용량 화면은 세션과 토큰 사용량을 보여 주지만[S14], 상용 유형이 제공하는 지출 한도, 부서별 할당, 한도 초과 시 자동 정지 같은 통제 기능은 아직 없습니다. 관리자는 작업공간 개수 한도를 사람마다 조정할 수 있지만[S13], 이것은 자원 한도이지 비용 한도가 아닙니다. 사내 sLLM을 쓰면 호출마다 과금되지는 않지만 GPU 장비와 가동률 비용은 그대로 남으므로, 시범 도입 단계에서는 OPENMARU AAS의 사용량 화면으로 업무별 사용량을 확인하고 LiteLLM 같은 모델 게이트웨이가 제공하는 키별 예산 설정을 함께 쓰는 구성을 권합니다. 이 구성이 맞지 않을 만큼 부서별 비용 배분이 엄격한 조직이라면, 비용 통제 기능을 갖춘 상용 유형과 비교한 뒤 결정하는 편이 안전합니다.

9장: 사용자와 관리자가 쓰는 주요 기능

8장의 비교가 유형 수준이었다면, 도입 뒤 실제로 마주할 화면은 더 구체적으로 확인해야 합니다. OPENMARU AAS의 화면 기능은 공개 매뉴얼 두 권에 정리돼 있습니다. 사용자 매뉴얼은 시작하기, 채팅, 오피스, AI 에이전트 설정, 스킬, 자동화, 기록과 설정의 7부로 나뉘고, 관리자 매뉴얼은 구성과 권한, 사용자와 조직, 작업공간 운영, 모델과 안전장치, 역할 템플릿·공지·요청, 스킬 허브 운영, 점검과 기록, 연동 서비스의 8부로 나뉩니다[S12][S13]. 도입을 검토하는 IT 담당자가 먼저 확인할 것은 업무 담당자가 매일 쓰는 화면과 IT 담당자가 통제에 쓰는 화면이 같은 제품 안에서 어떻게 이어지는가입니다. 매뉴얼의 부 구성을 따라 업무 담당자 화면, 스킬 수명 주기, 권한 4종과 작업공간 한도, 안전장치와 점검 화면을 차례로 보면, IT 담당자가 운영할 통제 수단이 충분한지도 화면 단위로 판단할 수 있습니다.

9.1 대화에서 승인과 예약 작업까지 이어지는 업무 담당자 화면

OPENMARU AAS는 사용자마다 AI 에이전트 작업공간을 하나씩 만들고, 그 작업공간에 설정·스킬·기억·파일을 저장합니다. 작업공간 안의 AI 에이전트 설정 묶음을 프로파일이라고 부르며, 처음에는 주 프로파일 하나가 있습니다. 업무 담당자가 쓰는 화면은 이 작업공간을 중심으로 대화 → 승인 → 예약 작업 순서로 이어집니다. 즉 한 번 말로 시킨 일이 사람의 허락을 거쳐 정해진 주기로 되풀이되는 과정이 한 제품 안에서 끝납니다. 아래 표는 이 흐름에 해당하는 사용자 매뉴얼의 쪽을 업무 순서대로 묶은 것입니다.

업무 흐름	화면	사용자 매뉴얼 쪽
일을 시킨다	채팅, 지난 대화와 새 대화, 파일, 문서 뷰어, 말로 입력과 읽어 주기	2.1–2.5
허락하고 답한다	승인과 질문에 답하기(받은편지함)	2.6
여럿이 함께 일한다	3D 오피스, 회의실과 TFT 방, 칸반	3.1–3.3
되풀이한다	예약 작업, 메신저 연결	6.1–6.2

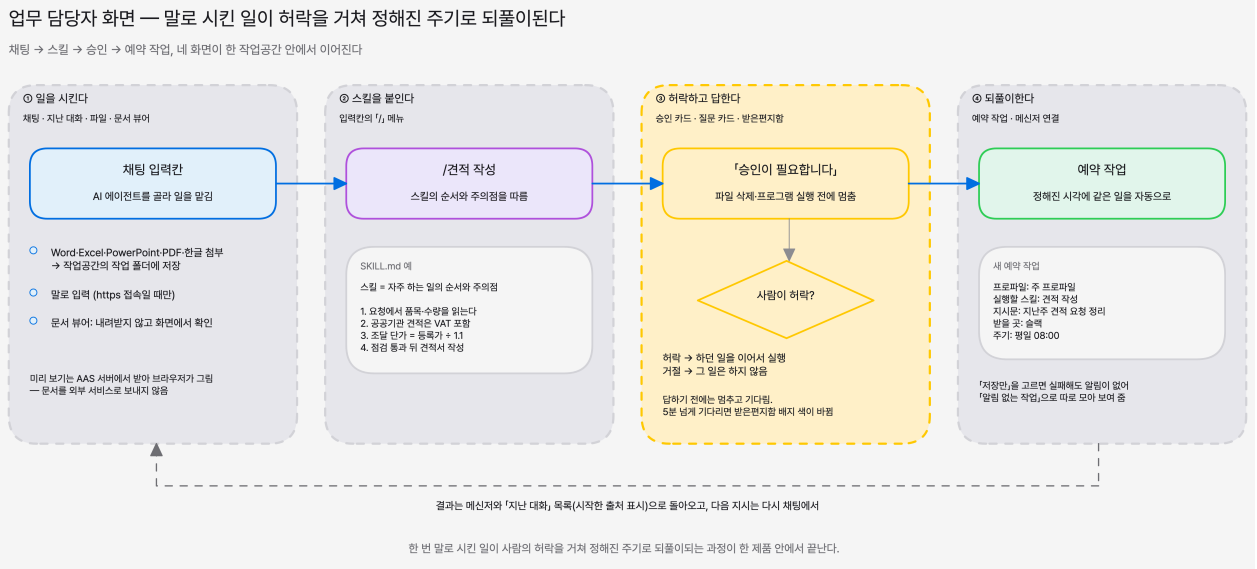


그림. 업무 담당자가 채팅으로 일을 시키고, 스킬을 붙이고, 승인 카드에 답하고, 예약 작업으로 되풀이하는 흐름

9.1.1 채팅·지난 대화·파일·문서 뷰어·음성 입력을 한곳에 모은 업무 지시 창

채팅과 지난 대화의 구분 채팅 화면에서는 왼쪽 목록에서 AI 에이전트를 골라 질문하거나 일을 맡깁니다. AI 에이전트는 답을 쓰는 동안 필요한 스킬과 도구를 직접 써서 자료를 찾거나 파일을 만들고, 입력칸에서 「/」 메뉴를 열면 스킬과 명령을 바로 부를 수 있습니다. 한 대화 안에서는 앞서 나눈 내용을 기억하므로, 이어 하던 일은 「지난 대화」 서랍에서 찾아 이어 쓰고 주제가 바뀌면 새 대화를 엽니다. 지난 대화 목록에는 대화를 시작한 출처가 함께 표시돼, 메신저나 예약 작업에서 시작된 대화도 같은 목록에서 구분해 볼 수 있습니다.

파일과 문서 뷰어 채팅에 붙인 Word, Excel, PowerPoint, PDF, 한글 문서는 작업공간의 작업 폴더에 저장되고, AI 에이전트가 그 내용을 읽어 요약하거나 정리합니다. 문서 뷰어는 채팅에 나온 파일을 내려받지 않고 화면 안에서 쪽 모양에 가깝게 열어 주며, 미리 보기는 AAS 서버에서 파일을 받아 브라우저 안에서 그리므로 문서 내용을 다른 외부 서비스로 보내지 않습니다[S12]. 따라서 AI 에이전트가 만든 보고서를 내려받기 전에 같은 화면에서 확인할 수 있습니다.

말로 입력과 읽어 주기 키보드 대신 말로 질문을 입력하면 받아쓴 글이 입력칸에 들어가고, 업무 담당자가 확인한 뒤 직접 보냅니다. 다만 이 기능은 설치에서 켜져 있고 보안 연결(https)로 접속했을 때만 버튼이 나타납니다. 이렇게 채팅·파일·문서 뷰어·음성 입력이 한 입력 창에 모여 있어, 업무 지시와 결과 확인을 다른 프로그램으로 옮기지 않고 끝낼 수 있습니다.

9.1.2 사람의 승인이 필요한 작업을 모으는 승인 받은편지함과 칸반·회의실·3D 오피스

승인 카드와 받은편지함 AI 에이전트는 파일을 지우거나 프로그램을 실행하는 것처럼 위험할 수 있는 일을 하기 전에 「승인이 필요합니다」 카드를 띄우고, 일을 이어 가려고 질문할 때는 질문 카드를 띄웁니다. 같은 내용이 상단 막대의 받은편지함에도 모이고, 답하기 전에는 AI 에이전트가 하던 일을 멈추고 기다립니다[S12]. 승인이나 질문이 5분 넘게 답을 기다리면 받은편지함 숫자 배지의 색이 바뀌어 오래 기다리는 요청을 따로 알려 줍니다. 이 구조 덕분에 사람의 허락이 필요한 일이 여러 대화창에 흩어지지 않고 한 목록에 모입니다.

3D 오피스·회의실·칸반 오피스 화면은 같은 부서의 AI 에이전트 여럿이 지금 무엇을 하는지 3D 사무실로 보여 주고, 같은 화면의 탭에서 칸반·회의실·TFT 방·배치로 넘어갑니다[S12]. 칸반은 일을 카드로 적어 AI 에이전트에게 맡기는 화면이며, 카드는 진행 중·막힘·검토·완료 칸을 옮겨 가다가 검토 칸에서 사람의 확인을 기다립니다. 회의실은 사람과 여러 AI 에이전트가 한 대화방에서 함께 일하는 곳이고, TFT 방은 특정 과제를 위해 목표와 구성원을 정해 만든 방입니다. 이 화면들 덕분에 채팅 화면을 지키지 않아도 맡긴 일의 상태를 칸반 칸과 오피스 화면에서 확인할 수 있습니다.

9.1.3 주기와 받을 곳을 화면에서 정하는 예약 작업과 메신저 알림

예약 작업의 설정 항목 예약 작업은 정해진 시각에 AI 에이전트가 같은 일을 자동으로 하는 기능입니다. 새 예약 작업을 만들 때는 실행할 프로파일, 작업 이름, 실행할 스킬, 지시문, 결과를 받을 곳, 주기를 고르고, 「이 작업이 하게 될 일」 문장을 읽어 내용이 맞는지 확인한 뒤 만듭니다[S12]. 주기는 매일, 평일, 매주, 매월, 시간 간격, 분 간격, cron 식 직접 입력 가운데 하나로 정하며, 간격이 5분보다 짧으면 모델 호출 비용을 알리는 경고가 나옵니다. 따라서 업무 담당자는 cron 식을 몰라도 화면의 선택지만으로 정기 업무를 등록할 수 있습니다.

결과를 받을 메신저 결과를 받을 곳 목록에는 그 프로파일에 연결해 켜 둔 메신저가 나오고, 연결할 수 있는 메신저는 텔레그램, 디스코드, 슬랙, Mattermost 입니다[S12]. 반면 「저장만」을 고르면 결과는 저장되지만 실패해

도 아무에게도 알리지 않으므로, 화면은 이런 작업을 「알림 없는 작업」으로 따로 모아 보여 줍니다. 이처럼 결과를 받을 곳과 실패 알림이 갈 곳을 만드는 단계에서 같이 정하도록 화면이 구성돼 있습니다.

9.2 스킬 만들기·게시 검토·허브 설치로 이어지는 스킬 수명 주기

스킬은 자주 하는 일의 순서와 주의점을 적어 둔 업무 방법이며, AI 에이전트는 같은 일을 맡을 때마다 그 순서를 따릅니다[S12]. OPENMARU AAS에서 스킬은 한 사람의 작업공간에서 만들어진 뒤 관리자의 게시 검토를 거쳐 사내 스킬 허브에 올라가고, 다른 사람이 허브에서 설치해 씁니다. 그래서 개인이 만든 업무 방법이 조직 자산이 되는 경로가 화면 세 곳에 나뉘어 있습니다. 아래 표는 그 세 단계를 권한과 함께 정리한 것입니다.

단계	화면	하는 사람	확인하는 것
만들기	스킬 › 내 스킬 만들기, Claude Code 연결	작업공간 주인·팀 관리자·관리자	스킬 문서와 스크립트, 연결도
게시 검토	스킬 허브 › 게시 요청	요청은 주인·팀 관리자·관리자, 승인·반려는 관리자	설명·스크립트·비밀값·내부 주소·이름과 버전·네임스페이스
허브 설치	스킬 › 스킬 허브	작업공간 주인·팀 관리자·관리자	공용·제한 구역, 새 버전

9.2.1 코딩 에이전트 연결로 시작하는 만드는 단계의 스킬 제작

스킬을 만드는 세 가지 방법 사용자 매뉴얼은 스킬을 만드는 방법으로 AI 에이전트에게 맡기기, 직접 쓰기, zip 올리기를 제시하고, 처음이거나 프로그램(스크립트)이 필요한 스킬이면 AI 에이전트에게 맡기는 방법을 권합니다[S12]. 업무 담당자가 할 일은 업무 순서와 판정 규칙을 문장으로 정하는 것이고, 스킬 문서와 스크립트를 쓰는 일은 에이전트가 맡습니다. 이 구조는 6장의 만드는 단계와 같으며, 스킬을 만들고 고치는 권한은 작업공간의 주인·팀 관리자·관리자에게 있습니다.

Claude Code 연결 코드가 많은 스킬은 「설정 › Claude Code 연결」 화면에서 시작할 수 있습니다. 내 PC의 Claude Code를 AAS에 연결하면 스킬을 PC에서 만들고 올리고, 실제 AI 에이전트로 시험해 고치는 일을 Claude Code 안에서 끝낼 수 있습니다[S12]. 사용자 매뉴얼은 이 기능을 Claude Code를 이미 쓰는 개발자를 위한 것으로 설명하므로, 시범 도입에서는 업무 담당자가 규칙을 말하고 IT 담당자나 개발자가 연결한 Claude Code가 스킬을 만드는 짝을 기본으로 삼습니다. 이때 Claude Code는 사용자 PC에서 본인 계정으로 실행되고, AAS는 그 계정의 인증 정보를 받거나 보관하지 않으며, 접속용 연결 토큰은 따로 발급합니다. 다만 연결 토큰으로 하는 일은 관리자의 연결이라도 관리자 권한을 쓰지 못하므로, 관리 작업은 브라우저 로그인으로만 합니다[S13]. 현재는 외부 코딩 에이전트인 Claude Code를 연결해 쓰고, 폐쇄망이면 사내 코딩 모델로 바꿀 수 있습니다. 스킬을 다 만든 뒤 매일 업무를 처리하는 일하는 단계는 조직 안 sLLM 위의 AI 에이전트가 맡습니다.

9.2.2 프로파일·모델·SOUL·비밀값·MCP 설정과 스킬 연결도

프로파일 설정 항목 프로파일은 AI 에이전트의 성격과 권한을 모아 둔 설정이며, 어떤 성격과 규칙으로 말할지, 어떤 모델을 쓸지, 어떤 외부 서비스에 접속할지가 여기에 들어 있습니다. 아래 표는 프로파일 상세에서 고치는 항목과 그 쓰임을 정리한 것입니다.

설정	쓰임	사용자 매뉴얼 쪽
프로파일과 모델	주 모델 확인, 모델 변경은 관리자에게 요청	4.2
성격(SOUL)·작업지침(AGENTS)	역할·말투·규칙, 일하는 순서와 보고 형식	4.3
비밀값과 인증 파일	외부 서비스 접속용 키를 넣고 지움 (값은 다시 보이지 않음)	4.4
외부 도구(MCP) 연결	일정·이슈 관리·문서 관리 시스템 같은 외부 서비스 연결	4.5

설정 항목별 주의 성격(SOUL)은 AI 에이전트가 모든 대화에서 따르는 기본 지시이고, 작업지침(AGENTS)은 일을 어떤 순서와 형식으로 처리할지 적는 글입니다. 비밀값은 채팅 창에 붙여 넣지 않고 비밀값 탭에만 넣으며, 주인과 관리자를 포함해 누구도 화면에서 값을 다시 볼 수 없습니다[S13]. MCP(Model Context Protocol) 서버는 넣은 키의 권한으로 동작하므로, 오피스나 회의실에서 다른 사람이 이 AI 에이전트에게 일을 시켜도 같은 권한을 씁니다. 그래서 사용자 매뉴얼은 개인 계정의 전체 권한 키 대신 필요한 범위만 가진 키를 쓰라고 안내합니다.

스킬 연결도 스킬 연결도는 스킬이 어디서 불리고 무엇을 부르는지 그림으로 보여 주는 화면이며, 스킬·예약 작업·스크립트·MCP 서버가 각각 노드로 나타납니다[S12]. 이 그림으로 스킬이 외부로 무엇을 호출하는지 점검하고, 같은 화면에서 스킬 문서(SKILL.md)를 고치거나 이전 버전으로 되돌립니다. 다만 그림은 스킬 문서·스크립트·예약 작업 설정을 읽어 만든 결과이므로, 변수로 조립하는 주소와 AI 에이전트가 그때그때 판단해 부르는 도구는 보이지 않을 수 있습니다.

9.3 관리자·팀 관리자·작업공간 주인·일반 사용자 4종 권한과 작업공간 한도

OPENMARU AAS의 권한은 관리자, 팀 관리자, 작업공간 주인, 일반 사용자 네 가지로 나뉘고, 한 사람이 둘 이상을 겸할 수 있습니다[S13]. 관리자는 사내 로그인(SSO) 서버의 관리자 그룹에 들거나 「사용자·권한」 화면에서 지정하고, 팀 관리자는 관리자가 부서마다 지정하며, 작업공간 주인은 작업공간을 만든 사람입니다. 일반 사용자는 사내 로그인의 사용자 그룹 구성원으로 자기 작업공간을 쓰며, 관리자가 다른 사람의 작업공간을 공유해 주면 그 작업공간을 보고 대화만 할 수 있습니다(아래 표의 「공유받은 사람」 열). 아래 표는 관리자 매뉴얼 1.2의 역할별 가능 작업을 줄여 옮긴 것이며, 팀 관리자 열은 팀원 작업공간에 대한 권한입니다[S13].

하는 일	관리자	팀 관리자	작업공간 주인	공유받은 사람
작업공간 보기, AI 에이전트와 대화	모든 작업공간	팀원 작업공간	자기 작업공간	공유받은 작업공간
성격·작업 지침·스킬·도구·MCP 바꾸기	가능	가능	가능	불가
메신저 연결, 비밀값 넣기·지우기	가능	가능	가능	불가

하는 일	관리자	팀 관리자	작업공간 주인	공유받은 사람
예약 작업 만들기와 바꾸기	가능	가능	가능	불가
스킬 허브에서 설치와 지우기	가능	가능	가능	불가
모델·제공자·게이트웨이·터미널 실행 위치 바꾸기	가능	불가	불가	불가
소유권 옮기기, 사람별 개수 한도	가능	불가	불가	불가
변경 기록 보기	전체	팀원 작업공간	자기 작업공간	불가

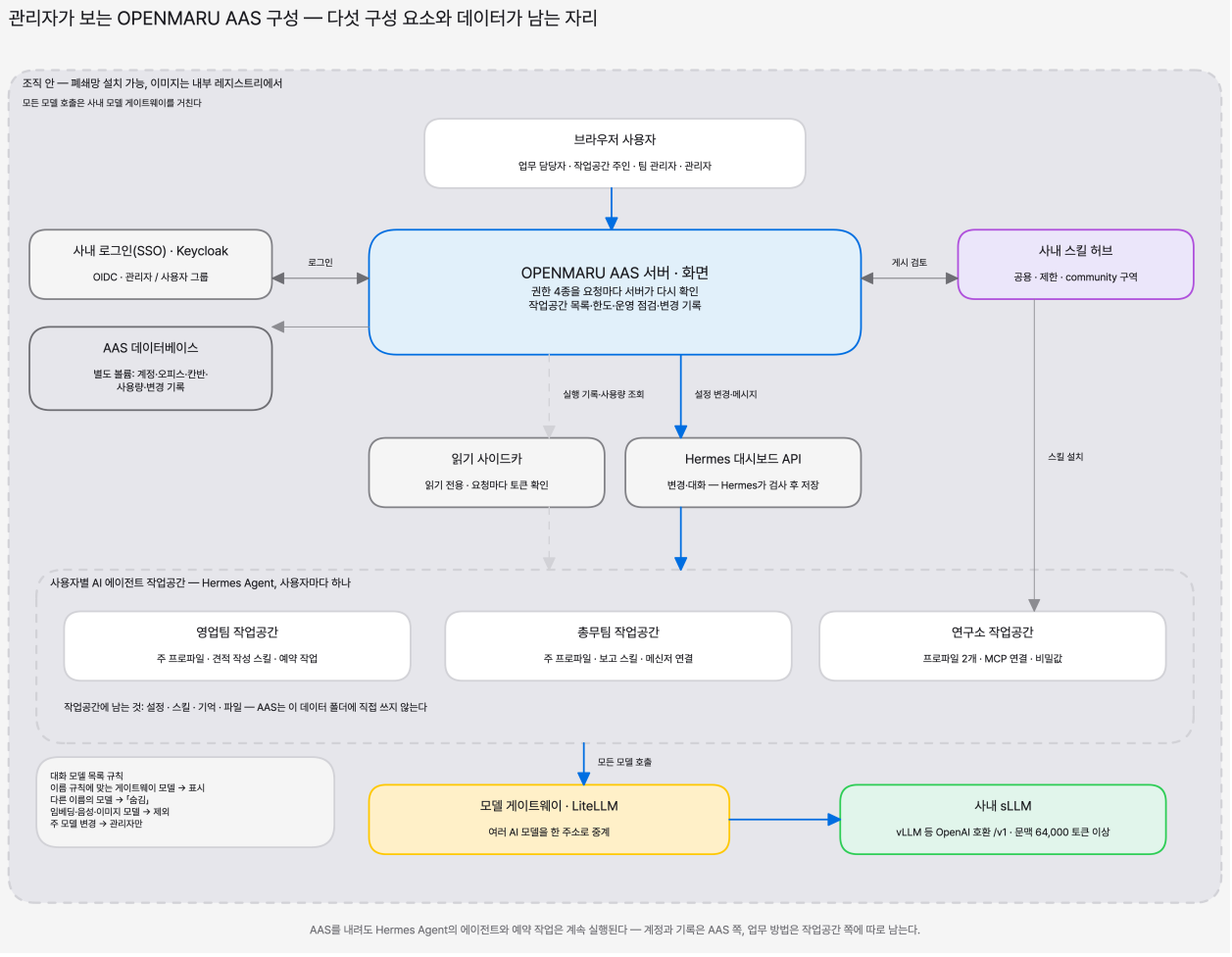


그림. AAS 서버와 화면, 사내 로그인(SSO), 모델 게이트웨이(LiteLLM), 스킬 허브, 사용자별 AI 에이전트 작업공간의 연결과 데이터가 남는 자리

9.3.1 사용자·부서·역할 템플릿과 작업공간 상태 관리

사용자와 부서 관리자가 「관리 › 사용자·권한」 화면에서 사람을 추가하거나 고치면 AAS가 SSO 서버에 직접 쓰므로, SSO 서버 관리 화면에서 계정을 따로 만들 필요가 없습니다[S13]. 부서는 공지 대상, 팀 관리자 지정, 3D 오피스 팀 묶기, 역할 템플릿 추천에 쓰이므로 관리자 매뉴얼은 첫날 부서 목록을 정해 두라고 권합니다. 권한 변경은 정해진 시간 안에 반영되며, 관리자에서 내린 사람은 서버에서 로그인이 바로 끊깁니다. 화면에서 메뉴를 가리는 것은 편의이고, 실제 권한은 서버가 모든 요청마다 다시 확인합니다.

역할 템플릿과 작업공간 운영 역할 템플릿은 마케팅·세일즈·연구소·엔지니어팀·총무·회계·인사·고객지원·기획·법무 열 가지 역할의 성격(SOUL) 글을 한국어와 영어로 하나씩, 모두 20개 기본으로 제공하며, 관리자는 이를 고치거나 회사에 맞는 템플릿을 새로 만듭니다[S13]. 작업공간 쪽에서는 관리자가 모든 사람의 작업공간을 한 목록으로 보고, 사람마다 만들 수 있는 개수 한도를 바꾸고, 주인이 없어진 작업공간의 새 주인을 정합니다. 작업공간 상태는 사용할 수 있음·준비 중·만드는 중·실패·멈춤·연결 끊김으로 표시되고, 멈춤에는 「오래 쓰지 않아 멈춤」 같은 이유가 붙습니다. 사용자를 삭제해도 그 사람의 작업공간 데이터는 남고 작업공간이 멈출 뿐이며, 새 주인은 관리자가 정합니다.

관리 항목	관리자 매뉴얼 부	관리자가 하는 일
사용자·부서·팀 관리자	2부	계정 추가·수정·사용 중지, 부서 목록, 팀 관리자 지정
작업공간 한도·상태	3부	개수 한도, 새 주인 지정, 멈춘 작업공간 다시 시작
역할 템플릿·공지·요청	5부	성격 템플릿, 부서별 공지, 모델 변경 같은 사용자 요청 처리

9.3.2 운영자가 정한 이름 규칙에 맞는 게이트웨이 모델만 보이는 대화 모델 목록

모델 게이트웨이와 노출 규칙 OPENMARU AAS의 AI 에이전트는 LiteLLM을 거쳐 AI 모델을 부르고, LiteLLM은 여러 AI 모델을 한 주소로 쓰게 해 주는 중계 서버입니다[S13]. 대화용 모델 목록에는 운영자가 정한 이름 규칙에 맞는 게이트웨이 모델만 보이고, 다른 이름의 모델은 「숨김」으로 세어져 목록에서 빠집니다. 보조 작업 전용 별칭과 임베딩·음성·이미지 모델도 대화용 선택 목록에서 빠집니다. 이 규칙은 모델별 승인 절차가 아니라 이름으로 거르는 방식이므로, 어떤 모델을 목록에 올릴지는 운영자가 LiteLLM에 모델을 등록하고 이름을 붙이는 단계에서 정해집니다.

모델 화면에서 보는 것 모델 화면은 LiteLLM이 내놓는 모델 목록과 프로파일마다 쓰는 모델을 함께 보여 주며, 폴백 설정 오류로 실패한 예약 작업 수와 어느 프로파일도 쓰지 않는 모델 수를 카드로 표시합니다[S13]. 이름 규칙 밖의 모델을 프로파일이 쓰고 있으면 「AAS 모델 아님」 경고가 붙고, 이런 모델에는 질문 가드레일이 적용되지 않을 수 있습니다. 주 모델을 바꾸는 일은 관리자만 하고, 작업공간 주인과 팀 관리자는 변경 요청을 보내며 그 요청은 관리자의 받은편지함에 모입니다. 공개 저장소 기준으로 AAS는 OpenAI 호환 주소 하나로 사내 vLLM·LiteLLM·Ollama 같은 모델 서버에 연결되고, 에이전트가 도구를 쓰려면 문맥 길이가 64,000 토큰 이상이어야 합니다[S14].

9.4 화면에서 끌 수 없는 질문 가드레일·위험 명령 차단과 사용량·변경 기록 점검

OPENMARU AAS는 AI 에이전트가 하는 일을 질문 가드레일과 위험한 명령 차단 두 곳에서 막고, 두 안전장치 모두 관리자 화면에서 켜거나 끄는 설정이 없습니다[S13]. 질문 가드레일은 사용자가 보낸 질문이 정책에 어긋

날 때 모델을 부르지 않고, 이 동작은 채팅·메신저 연결·예약 작업에 모두 같습니다. 위험한 명령 차단은 AI 에이전트가 자기 데이터 폴더를 통째로 지우는 명령을 실행하지 못하게 하며, 사용자가 승인 카드에서 허락해도, 승인 기능을 꺼도 막힙니다. 아래 표는 두 안전장치와 IT 담당자가 운영 상태를 확인하는 점검 화면을 함께 정리한 것입니다[S13].

구분	이름	하는 일	한계 또는 범위
안전장치	질문 가드레일	정책에 어긋난 질문이면 모델을 부르지 않음	지금 보낸 사람의 글만 검사, 서비스 장애 동안에는 통과, 이름 규칙 밖 모델은 거치지 않음
안전장치	위험한 명령 차단	루트·홈·데이터 폴더·프로파일 통째 삭제를 막음	작업 폴더 안 하위 파일 삭제는 막지 않고 승인 카드로 물음
점검 화면	대시보드·운영 점검	작업공간 준비, 실패·멈춘 작업, 문제 있는 프로파일, 만료된 키	관리자
점검 화면	사용량·변경 기록	세션·토큰 수, 누가 언제 무엇을 바꿨는지	관리자는 전체, 주인·팀 관리자는 자기 범위

9.4.1 스킬 허브 구역과 관리자 게시 검토 절차

공용·제한·community 구역 스킬 허브는 사내에서 쓰는 스킬 패키지를 모아 두고 프로파일에 설치하게 하는 곳이며, 패키지는 네임스페이스(이름 구역) 단위로 나뉩니다[S13]. 「공용」은 모든 작업공간이 받을 수 있고, 「제한」은 관리자가 작업공간마다 허용해야 받을 수 있으며, community는 외부에 공개된 스킬 가운데 라이선스를 확인한 것을 복사본으로 넣은 읽기 전용 구역입니다. 공용과 제한은 가격이 아니라 받을 수 있는 범위를 나타내며 결제와 이어져 있지 않습니다. 누가 언제 허용했는지는 변경 기록에 남습니다.

게시 요청 검토 단계 작업공간 주인·팀 관리자·관리자가 게시를 요청하면 서버가 먼저 스킬 폴더를 검사하고, 키·토큰 모양의 문자열이 있거나 파일 한도를 넘거나 SKILL.md가 없으면 요청 자체를 만들지 않습니다[S13]. 그다음 관리자가 묶음(zip)을 내려받아 내용을 확인하고 네임스페이스를 골라 승인하거나 반려합니다. 관리자 매뉴얼의 승인 전 확인 항목은 다음 표와 같습니다.

확인 항목	보는 내용
게시 내용	SKILL.md 설명이 실제 하는 일과 맞는지, 스크립트에 파일 삭제·외부 전송 같은 의도하지 않은 동작이 없는지
비밀값	키·토큰·비밀번호 같은 글자가 없는지
내부 주소	사내 도메인이나 사설 IP 경고가 있으면 「제한」에만 게시
이름과 버전·네임스페이스	기존 패키지를 의도치 않게 덮어쓰지 않는지, 공용과 제한 중 알맞은 쪽인지

게시한 버전은 되돌리거나 지울 수 없고, 잘못 올렸으면 관리자가 허브에서 숨깁니다[S12]. 결국 개인 작업공간의 스킬이 다른 부서로 퍼지기 전에 서버 검사와 관리자 검토라는 두 번의 확인을 거치게 됩니다.

9.4.2 사용량·변경 기록·연동 상태로 보는 조직 전체 에이전트 운영 상태

운영 점검 순서 관리자 매뉴얼은 대시보드 → 예약 작업 → 메신저 연결 → API 키·토큰 순서의 점검 절차를 제시하고, 하루 한 번 또는 주 한 번 대시보드의 숫자를 보라고 권합니다[S13]. 대시보드의 숫자 카드는 작업공간 준비 수, 실패·멈춘 작업, 메신저 연결에 문제가 있는 프로파일, 만료됐거나 곧 만료되는 키·토큰을 보여 주고, 0이 아닌 카드를 누르면 해당 화면으로 이어집니다. 「운영 정보」 한 줄에는 LiteLLM과 스킬 허브의 연결 상태가 나오므로, 모델 서버나 허브가 끊긴 상황을 같은 화면에서 확인합니다. API 키·토큰 화면은 값을 보여 주지 않고 이름·위치·만료일·빈 값 여부, 같은 값이 여러 곳에 복사됐는지만 표시합니다[S12].

사용량과 변경 기록 사용량 화면은 작업공간별 세션 수·메시지 수·도구 호출 수·입력과 출력 토큰 수를 보여 주고, 대시보드의 추이 그래프는 전체 작업공간 사용량의 30일 합계를 보여 줍니다[S13]. 변경 기록은 누가 언제 무엇을 바꿨는지 남기며, 관리자는 모든 작업공간과 접속 기록을 보고 작업공간 주인과 팀 관리자는 자기 범위만 봅니다. 변경 기록은 비밀값과 글 내용을 기록하지 않고 바꾼 사실만 남깁니다. 연동 서비스 쪽에서는 SSO, 모델 서버, 스킬 허브, 받아쓰기·읽어 주기 서버, 위키 연결 가운데 무엇이 끊기면 사용자가 어떤 증상을 겪는지와 관리자가 확인할 화면을 관리자 매뉴얼 8부가 표로 정리합니다.

점검 대상	화면	드러나는 위험
예약 작업	자동화 › 예약 작업	마지막 실행 실패, 멈춘 작업, 알림 없는 작업
메신저 연결	자동화 › 메신저 연결	게이트웨이 중지로 실행되지 않는 예약 작업
API 키·토큰	설정 › API 키·토큰	만료·만료 임박, 빈 값, 여러 곳에 복사된 값
사용량·변경 기록	기록 › 사용량, 기록 › 변경 기록	갑자기 늘어난 사용량, 원인이 된 설정 변경

따라서 IT 담당자는 권한 4종으로 누가 무엇을 바꿀 수 있는지 정하고, 끌 수 없는 안전장치 두 가지 위에서, 대시보드와 기록 화면으로 회사 전체 에이전트의 운영 상태를 확인하는 통제 체계를 갖게 됩니다.

IT·보안 담당이 도입 전에 묻는 질문

에이전트가 업무 애플리케이션 곳곳에 들어오면 에이전트 플랫폼도 조직의 업무 시스템 목록에 오릅니다. IT·보안 담당이 도입 전에 묻는 질문은 대개 세 가지로 모입니다. 도입하면 운영 중인 에이전트 플랫폼의 설정이나 플러그인을 바꿔야 하는지, 모델 호출과 업무 자료가 조직 밖으로 나가는지, 그리고 누가 언제 무엇을 바꿨는지 나중에 추적할 수 있는지입니다. 첫 번째 질문은 7.3절의 공식 API 경로가, 두 번째 질문은 4장의 사내 모델 게이트웨이 구성이, 세 번째 질문은 이 절의 권한 4종과 변경 기록이 답합니다. 도입 검토 회의에서는 이 세 질문과 답하는 절을 한 장에 묶어 두면, 보안 검토에서 같은 설명을 되풀이하지 않아도 됩니다.

10장: OPENMARU AAS가 있을 때와 없을 때의 업무 비교

9장에서 본 화면들이 실제 업무에 들어오면 달라지는 일은 두 쪽에서 나타납니다. 업무 담당자 쪽에서는 규칙을 개발자에게 다시 설명하고 결과를 다른 시스템에 옮겨 적던 수작업이 줄고, IT 담당자 쪽에서는 사람마다 흩어져 있던 에이전트·자격증명·예약 작업이 한 화면에서 드러납니다. 2장에서 본 것처럼 생성형 AI 프로젝트는 2025년 말까지 최소 30% 가 개념 검증 뒤 중단되고, 에이전틱 AI 프로젝트는 2027년 말까지 40% 이상이 비용 상승·불분명한 가치·위험 통제 부족으로 취소될 것이라는 전망이 있습니다[S3][S4]. 그래서 도입 효과는 기술 시연보다 결재 문서에 적을 수 있는 업무 장면과 지표로 설명해야 합니다. 업무 장면 8가지의 전후 비교와 업무 담당자·조직 쪽 변화를 정리하면, 시범 도입에서 도입 조직이 직접 채울 효과 지표 틀까지 이어집니다.

10.1 업무 장면 8가지로 본 OPENMARU AAS가 있을 때와 없을 때

업무 장면 8가지는 만드는 단계와 일하는 단계에 걸쳐 고른 것으로, 앞의 4가지는 에이전트를 만들고, 일을 시키고, 결과를 확인하고, 자료가 어디로 가는지 보는 장면이며, 뒤의 4가지는 규칙 변경·견적 요청·반복 보고·담당자 교체처럼 업무가 계속 이어질 때 생기는 장면입니다. 두 표는 결재 문서에 그대로 옮길 수 있도록 「없을 때」와 「있을 때」를 같은 행에 나란히 적었습니다. 각 행의 「있을 때」는 OPENMARU AAS 사용자·관리자 매뉴얼에 적힌 화면 기능과 대응합니다[S12][S13].

10.1.1 에이전트 만들기·일 시키기·결과 확인 장면의 전후 비교

만드는 단계와 일 시키는 장면 에이전트를 만드는 장면에서 가장 크게 달라지는 것은 업무 방법을 누가 적느냐입니다. OPENMARU AAS가 없으면 업무 담당자가 개발자에게 요청하고 기다려야 하지만, AAS 에서는 업무 담당자가 순서와 판정 규칙을 말로 정하고 AI 에이전트나 연결된 코딩 에이전트가 스킬 문서와 스크립트를 만듭니다[S12]. 일을 시키는 장면에서는 메신저 봇마다 따로 등록하던 일이 채팅 화면 한 곳으로 모이고, 승인과 질문이 일반 메시지 사이에 섞이지 않고 승인 카드와 받은편지함으로 따로 모이며, 아래 표는 이 네 장면을 도입 전후로 나란히 놓은 것입니다.

장면	OPENMARU AAS가 없을 때	OPENMARU AAS가 있을 때
1. 업무 방법(스킬) 만들기	업무 담당자가 개발자에게 규칙을 설명하고 구현을 기다린다	업무 담당자가 말로 정의하면 에이전트가 스킬로 만들고, 사내 스킬 허브에서 이미 있는 스킬을 받는다
2. 에이전트에게 일 시키기	메신저 봇마다 따로 등록하고 승인·질문이 메시지에 섞인다	한 화면에서 에이전트를 고르고, 승인·질문 카드에 바로 답한다
3. 결과 확인	만든 파일을 내려받아 다른 프로그램에서 열어 본다	문서 뷰어로 화면 안에서 열고, 칸반 검토 칸에서 확인한다
4. 자료와 모델	직원이 외부 AI 서비스에 자료를 붙여 넣는다	조직 안 sLLM으로 동작하고, 운영자가 정한 이름 규칙에 맞는 게이트웨이 모델만 대화 목록에 보인다

결과 확인과 자료 보안 장면 결과를 확인하는 장면에서는 AI 에이전트가 만든 문서를 문서 뷰어가 화면 안에서 열어 주고, 미리 보기를 브라우저 안에서 그리므로 문서 내용이 다른 외부 서비스로 나가지 않습니다[S12]. 자료

보안 장면의 차이는 모델 연결 방식에서 나오는데, AAS의 AI 에이전트는 모델 게이트웨이인 LiteLLM을 거쳐 모델을 부르고, 대화용 목록에는 운영자가 정한 이름 규칙에 맞는 게이트웨이 모델만 보입니다[S13]. 따라서 조직 안 sLLM만 게이트웨이에 등록해 두면, 일하는 단계에서 고객·금액 자료가 조직 밖 LLM으로 나가지 않는 구성을 화면 설정으로 유지할 수 있습니다.

10.1.2 규칙 변경·견적 요청·반복 보고 장면의 전후 비교

업무가 이어질 때 생기는 장면 뒤의 네 장면은 에이전트를 한 번 만든 뒤 업무가 계속될 때 드러나는 차이입니다. 규칙이 바뀌면 OPENMARU AAS가 없는 조직에서는 개발자가 코드를 고치고 다시 배포해야 하지만, AAS 에서는 업무 담당자가 바뀐 규칙을 말로 알려 주고 코딩 에이전트가 규칙 파일·점검 코드·시험을 함께 고칩니다. 견적 요청은 6장의 사례처럼 한 줄 요청으로 시작해 CRM 견적 생성, PDF 저장, 영업기회 일지 기록까지 이어집니다. 반복 보고는 예약 작업 화면에서 주기와 결과를 받을 메신저를 정해 두면 정해진 시각에 실행됩니다[S12].

장면	OPENMARU AAS가 없을 때	OPENMARU AAS가 있을 때
5. 업무 규칙 변경	개발자에게 다시 설명하고 수정·배포를 기다린다	업무 담당자가 말로 고치고, 새 규칙을 지난 견적 전체에 다시 대어 통과·경고·차단 분포를 본다
6. 견적 요청 한 건	규칙 문서 확인, CRM 입력, PDF 저장, 일지 기록을 사람이 차례로 한다	「견적」 한 줄로 시작해 사람은 질문에 답하고 품목과 최종 내용을 확인한다
7. 반복 보고	매번 사람이 요청하고, 예약 실행이 실패해도 뒤늦게 안다	예약 작업이 정해진 주기로 실행하고, 실패·멈춤·알림 없는 작업이 한 화면에 모인다
8. 담당자 교체	업무 방법이 개인 PC와 기억에 남아 사람과 함께 떠난다	스킬과 예약 작업이 작업공간에 남고, 관리자가 새 주인을 정한다

규칙 변경 장면의 점검 방식 규칙 변경 장면에서 의사결정권자가 볼 차이는 바꾼 규칙이 과거 업무에 어떤 영향을 주는지 바로 확인된다는 점입니다. 견적 작성 스킬은 새 규칙을 지난 견적 전체에 다시 대어 보고, 과거 견적 가운데 통과·경고·차단이 몇 건인지 보여 줍니다. 견적을 만들기 전과 만든 뒤 두 번 점검하므로, 공공기관 견적인데 VAT 별도로 작성된 견적이나 조달 견적에 할인이 걸린 견적은 만들기 전에 멈춥니다. 이처럼 규칙을 고치는 사람과 규칙을 지키는 코드가 같은 스킬 안에서 함께 바뀌어, 규칙 문서와 실제 처리 사이의 불일치가 줄어듭니다.

10.2 업무 담당자 쪽 변화 — 규칙 재설명과 결과 재입력 수작업의 축소

업무 담당자 쪽에서 줄어드는 수작업은 두 가지입니다. 첫째는 같은 규칙을 개발자나 새 담당자에게 되풀이해 설명하는 일이고, 둘째는 AI가 만든 결과를 업무 시스템과 문서 저장소에 손으로 옮겨 적는 일입니다.
OPENMARU AAS 에서는 규칙이 스킬 안의 규칙 파일과 점검 코드로 남고, 결과 저장과 기록은 스킬의 처리 단계에 들어갑니다. 아래에서는 6장의 견적 요청 한 건을 예로 사람 손이 가는 단계를 전후로 셉니다. 시간 수치는 조직마다 다르므로 적지 않고, 10.4의 지표 틀에서 도입 조직이 직접 재도록 남겨 둡니다.

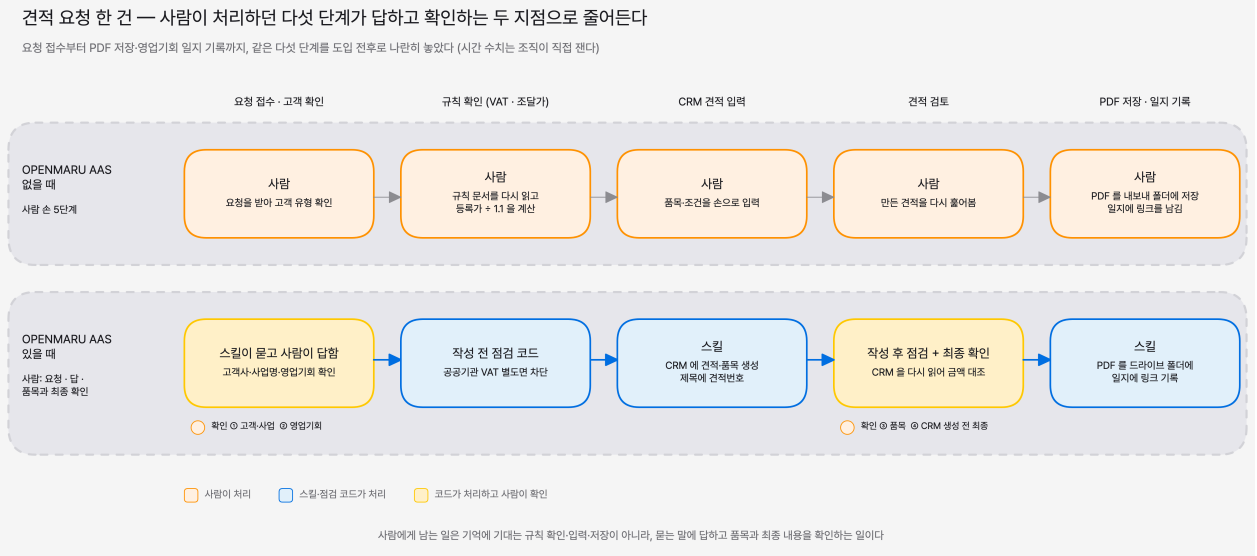


그림. 견적 요청 한 건이 요청 접수부터 PDF 저장·영업기회 일지 기록까지 가는 동안 사람 손이 가는 단계의 도입 전후 비교

10.2.1 견적 요청 한 건의 처리 흐름 전후 비교

도입 전 처리 단계 OPENMARU AAS가 없을 때 견적 요청 한 건은 대부분 사람이 처리합니다. 담당자는 요청을 받아 고객 유형을 확인하고, 규칙 문서를 찾아 VAT 포함 여부와 조달 등록가 계산 규칙을 다시 읽은 뒤 CRM에 품목을 입력합니다. 예를 들어 조달 등록가는 VAT 포함가이므로 품목 단가를 「등록가 ÷ 1.1」로 계산해야 하고, 공공기관 견적은 VAT 포함으로 작성해야 합니다. 담당자는 이렇게 만든 견적을 다시 검토하고, PDF로 내보내 공유 폴더에 저장하고, 영업기회 일지에 링크를 남기는 일까지 사람이 차례로 합니다. 규칙 확인·입력·검토·저장·기록이 모두 사람의 기억과 손에 달려 있습니다.

도입 후 처리 단계 OPENMARU AAS가 있을 때는 업무 담당자가 채팅에 「견적」 한 줄을 보내면 스킬이 고객사와 사업명을 묻고, 기존 영업기회를 찾아 확인을 받습니다. 품목 확정과 최종 확인을 사람이 승인하면, 스킬이 CRM 견적을 만들고 PDF를 저장하고 영업기회 일지에 링크를 기록합니다. 이 과정에서 판단은 미리 만든 코드가 하고 모델은 요청의 뜻을 읽어 정해진 처리를 실행하므로, AAS 채팅에서 견적 PDF 요청은 모델이 도구를 직접 부르지 않고 처리됩니다. 따라서 사람에게 남는 단계는 요청, 질문에 대한 답, 품목과 최종 내용 확인입니다.

처리 단계	도입 전 담당	도입 후 담당
요청 접수와 고객·영업기회 확인	사람	스킬이 묻고 사람이 답한다
규칙 확인(VAT·조달 등록가)	사람이 규칙 문서를 읽는다	작성 전 점검 코드
CRM 견적 입력	사람	스킬
견적 검토	사람	작성 후 점검 코드와 사람의 최종 확인
PDF 저장과 일지 기록	사람	스킬

10.2.2 담당자 교체 뒤에도 이어지는 스킬과 예약 업무

작업공간에 남는 업무 방법 OPENMARU AAS에서 스킬과 예약 작업은 사람 개인의 PC가 아니라 프로파일과 작업공간에 저장됩니다. 사용자를 삭제해도 그 사람의 작업공간 데이터는 남고 작업공간은 멈출 뿐이며, 새 주인은 관리자가 정합니다[S13]. 관리자는 작업공간 소유권을 다른 사람에게 옮길 수도 있습니다. 담당자가 바뀌어도 업무 규칙이 담긴 스킬, 정해진 주기로 실행되던 예약 작업, 연결해 둔 메신저와 외부 도구 설정이 그대로 이어집니다.

인수인계 방식의 전후 비교 도입 전에는 인수인계가 문서와 구두 설명에 의존하므로, 문서에 없던 작성 관행은 담당자와 함께 사라지기 쉽습니다. 반면 도입 후에는 새 담당자가 스킬 연결도에서 스킬이 어디서 불리고 무엇을 부르는지 보고, 변경 기록에서 누가 언제 설정을 바꿨는지 확인합니다[S12]. 스킬을 사내 스킬 허브에 게시해 두었다면 다른 부서나 다른 작업공간에서 같은 스킬을 설치해 이어 쓸 수도 있습니다.

10.3 IT 담당자와 조직 쪽 변화 — 흩어진 에이전트·자격증명·예약 작업의 한 화면 관리

IT 담당자 쪽 변화는 위험이 드러나는 방식에서 나타납니다. 에이전트를 사람마다 따로 띄우면 예약 작업의 실패 알림이 갈 곳 없이 사라지고, 같은 토큰이 여러 곳에 복사돼 만료를 모르고, 업무 방법이 개인 작업공간에만 남습니다. OPENMARU AAS는 이런 상태를 숫자 카드와 목록으로 보여 주는 화면을 갖추고 있으며, 관리자는 대시보드에서 전체 작업공간의 문제 수를 보고 해당 화면으로 넘어갑니다[S13]. 아래 표는 관리 대상별로 도입 전후를 비교한 것입니다.

관리 대상	도입 전	도입 후 화면
예약 작업	실패하거나 멈춰도 뒤늦게 안다	마지막 실행 실패·멈춘 작업·알림 없는 작업 보기
자격증명	같은 토큰이 여러 곳에 복사돼 만료를 모른다	값은 숨기고 만료·빈 값·복사된 값·쓰는 곳 표시
사용량	에이전트별 사용량을 따로 모아야 안다	작업공간별 세션·도구 호출·토큰 수와 30일 추이
변경 기록	누가 설정을 바꿨는지 추적하기 어렵다	누가 언제 무엇을 바꿨는지 기록(비밀값과 글 내용은 기록하지 않음)

10.3.1 실패 알림이 갈 곳 없는 예약 작업과 멈춘 작업의 발견

예약 작업 화면이 드러내는 위험 유형 예약 작업은 결과를 받을 곳을 「저장만」으로 정하면 실패해도 아무에게도 알리지 않고, 프로파일의 게이트웨이가 중지돼 있으면 실패 기록 없이 아예 실행되지 않습니다[S12]. OPENMARU AAS의 예약 작업 화면은 이런 작업을 「알림 없는 작업」과 「멈춘 작업」 보기로 따로 모으고, 멈춘 작업에는 추정 원인과 지난 시간을 함께 보여 줍니다. 모델 화면은 폴백 설정 오류로 실패한 예약 작업 수를 카드로 표시합니다[S13]. 화면이 드러내는 위험 유형은 다음과 같습니다.

- 마지막 실행 실패 작업과 연속 실패 횟수
- 실패해도 알림이 가지 않는 「저장만」 작업
- 게이트웨이 중지로 실행되지 않는 멈춘 작업
- 폴백 설정 오류로 실패한 예약 작업
- 만료됐거나 곧 만료되는 키·토큰과 여러 곳에 복사된 값

점검 주기와 조치 경로 관리자 매뉴얼은 하루 한 번 또는 주 한 번 대시보드의 숫자를 보고, 0이 아닌 카드가 있으면 예약 작업 → 메신저 연결 → API 키·토큰 순서로 원인을 찾으라고 안내합니다[S13]. 실패한 작업의 서랍에서는 마지막 오류 글을 복사해 관리자에게 넘기거나, 「대화로 조사」를 눌러 AI 에이전트와 함께 원인을 찾을 수 있습니다. 사람이 뒤늦게 발견하던 문제가 정해진 점검 순서 안에서 드러납니다.

화면이 보여 주는 것과 조직이 정할 것

이런 관측 화면이 하는 일은 사람이 보지 않는 시각에 실행된 업무를 숫자와 목록으로 드러내는 것입니다. 2장에서 본 것처럼 부족한 위험 통제는 에이전틱 AI 프로젝트가 중단되는 원인 가운데 하나이므로, 상태를 먼저 보이게 하는 일은 위험 통제의 첫 단계에 해당합니다. 다만 화면은 실행 결과를 보여 줄 뿐이고, 실패율이 어느 수준을 넘으면 그 업무를 사람에게 되돌릴지는 조직이 정해야 합니다. 프로파일을 부서나 업무 단위로 나눠 두면 담당자가 아닌 팀 관리자도 자기 팀 에이전트가 정해진 일을 하고 있는지 확인하기 쉬워집니다. 담당자가 아닌 사람도 에이전트의 상태를 읽을 수 있어야 시범 단계를 지나 업무에 정착할 수 있습니다.

10.3.2 개인에게 남던 업무 방법이 공유 스킬로 바뀌는 조직 자산화

개인 작업에서 부서 공유까지의 경로 OPENMARU AAS에서 업무 방법이 조직 자산이 되는 경로는 개인 작업공간의 스킬 → 게시 요청 → 관리자 검토 → 사내 스킬 허브 → 다른 작업공간 설치 순서입니다[S13]. 게시 요청 단계에서는 서버가 비밀값으로 보이는 문자열과 파일 한도를 먼저 검사하고, 관리자는 스크립트에 의도하지 않은 동작이 없는지와 내부 주소가 들어 있는지를 확인한 뒤 「공용」 또는 「제한」 구역을 골라 승인합니다. 허브에 올라간 스킬은 다른 부서의 작업공간 주인이 설치하고, 새 버전이 나오면 「새 버전 있음」 목록에 표시됩니다.

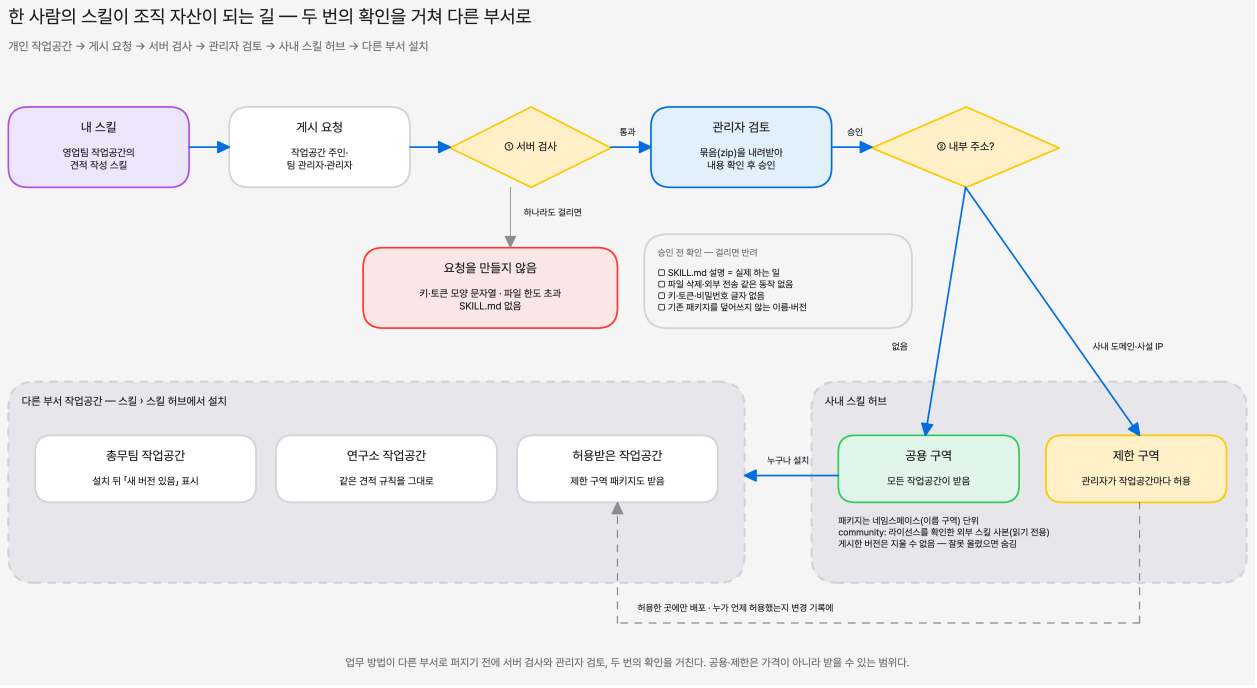


그림. 개인 작업공간의 스킬이 게시 요청과 관리자 검토를 거쳐 사내 스킬 허브의 공용·제한 구역에 올라가고 다른 부서 작업공간에 설치되는 경로

자산화가 조직에 주는 의미 잘 만든 업무 방법이 한 사람의 PC에 묻히면 다른 부서는 같은 일을 처음부터 다시 만들어야 합니다. 반면 사내 스킬 허브에 올라간 스킬은 관리자 검토를 거친 상태로 여러 부서가 받아 쓰고, 「제

한」 구역을 쓰면 허용한 작업공간에만 배포됩니다[S13]. 따라서 IT 담당자는 어느 스킬이 어느 범위까지 퍼져 있는지 허브 화면에서 확인하면서 조직 단위로 업무 방법을 관리할 수 있습니다.

10.4 시범 도입 효과를 결재 문서에 적는 지표 — 처리 시간·결과 품질·운영 통제

에이전틱 AI 프로젝트가 취소되는 원인으로 꼽힌 비용 상승·불분명한 가치·위험 통제 부족은 결재 문서가 미리 답해야 하는 세 질문이기도 합니다[S4]. 그래서 시범 도입 효과는 처리 시간·결과 품질·운영 통제 세 범주로 적는 것이 알맞습니다. 처리 시간은 가치를, 결과 품질은 업무 규칙을 지켰는지를, 운영 통제는 위험 통제 수단이 실제로 작동하는지를 보여 줍니다. 수치는 도입 조직이 시범 도입 전후에 같은 업무를 같은 방법으로 재서 채우고, 첫 업무는 가장 단순한 해법부터 시작하라는 원칙에 따라 범위가 좁은 업무 하나로 고릅니다[S11].

10.4.1 도입 조직이 자기 수치를 채우는 기대 효과 서술 틀

빈칸 표 사용 방법 아래 표의 빈칸은 도입 조직이 시범 도입 전과 시범 도입 기간에 같은 길이의 기간을 정해 같은 업무 하나를 대상으로 재서 채웁니다. 「도입 전」 칸은 현재 처리 방식으로 잰 값이고, 「시범 도입 후」 칸은 OPENMARU AAS의 스킬과 예약 작업으로 처리한 값입니다. 예를 들어 견적 작성을 첫 업무로 고른 조직이라면 「건당 처리 시간」은 요청 접수부터 PDF 저장·일지 기록까지의 시간이 됩니다. 채운 표는 12장의 첫 업무 선정과 결재 점검표로 그대로 이어집니다.

범주	지표	도입 전	시범 도입 후	변화
처리 시간	건당 처리 시간(요청 접수 → 결과 저장·기록)	( )	( )	( )
처리 시간	건당 사람 손이 가는 단계 수	( )	( )	( )
결과 품질	규칙 위반으로 다시 작성한 건수	( )	( )	( )
결과 품질	작성 전 점검에서 멈춘(차단된) 건수	해당 없음	( )	—
운영 통제	사람 확인(승인·질문 응답) 횟수	( )	( )	( )
운영 통제	실패를 알게 되기까지 걸린 시간	( )	( )	( )
운영 통제	만료·복사된 자격증명 건수	( )	( )	( )

10.4.2 처리 시간·규칙 위반 차단 건수·사람 확인 횟수의 측정 방법

지표별 측정 근거 화면 지표는 도입 전과 시범 도입 후에 같은 정의로 재야 비교할 수 있습니다. OPENMARU AAS 쪽 값은 이미 있는 화면에서 읽을 수 있으므로 별도 측정 도구를 만들 필요가 없습니다. 지난 대화에는 대화를 시작한 시각과 출처가 남고, 사용량 화면에는 작업공간별 세션 수·메시지 수·도구 호출 수가 나오며, 받은편지

함에는 승인과 질문이 모입니다[S12]. 반면 도입 전 값은 현재 처리 기록에서 표본을 뽑아 재야 하므로, 시범 도입을 시작하기 전에 표본 건수와 기간을 먼저 정해 둡니다.

지표	무엇을 세는가	도입 전 측정	시범 도입 후 측정
처리 시간	요청 접수부터 결과 저장·기록까지	처리 기록의 요청·완료 시각 표본	대화 기록의 요청 시각과 결과 저장 시각
규칙 위반 차단 건수	규칙을 어겨 다시 작성했거나 만들기 전에 멈춘 건수	반려·재작성 이력	작성 전·후 점검 결과, 새 규칙을 지난 업무에 다시 대어 본 통과·경고·차단 분포
사람 확인 횟수	사람이 승인하거나 질문에 답한 횟수	검토·결재 단계 수	받은편지함의 승인·질문 카드 수
실패 인지 시간	예약 실행 실패부터 담당자가 알기까지	장애 보고 시각	예약 작업 화면의 실패 시각과 확인 시각

측정 결과를 읽는 방법 세 범주는 함께 읽어야 합니다. 처리 시간이 줄었는데 규칙 위반 차단 건수가 늘었다면 점검 코드가 그동안 놓치던 위반을 잡기 시작한 것일 수 있고, 사람 확인 횟수가 줄었다면 승인 단계가 지나치게 빠진 것은 아닌지 확인해야 합니다. 그래서 결재 문서에는 세 범주의 변화를 함께 적고, 각 변화의 원인을 화면 근거로 붙입니다. 이 표가 채워지면 의사결정권자는 시범 도입한 업무 하나의 효과를 가치와 위험 통제 양쪽에서 판단할 수 있습니다.

11장: 도입 조직과 개발자 입장에서 본 Apache-2.0 오픈소스

10장의 효과 지표와 함께 결재 문서에 반드시 오르는 항목은 라이선스입니다. OPENMARU AAS의 소스 코드는 공개 저장소 github.com/openmaru-io/ai-agent-studio에 Apache License 2.0(이하 Apache-2.0)으로 공개돼 있습니다[S14]. 도입을 검토하는 조직의 법무·보안 담당자는 「오픈소스」라는 말만으로는 판단할 수 없고, 코드를 어디까지 고쳐 쓸 수 있는지, 회사 이름과 로고도 함께 쓸 수 있는지, 함께 들어 있는 제3자 구성요소의 라이선스는 무엇인지, 공개되지 않은 부분은 어디인지를 따로 확인해야 합니다. 에이전트를 도입하는 조직과 그 위에서 스킬을 만드는 개발자의 입장에서 네 가지 사실을 차례로 확인하면, 라이선스와 공급사 종속 위험도 결재 항목으로 판단할 수 있습니다.

11.1 코드는 Apache-2.0, 이름·로고는 상표 정책 — 도입 조직의 설치·수정·재배포 권리

OPENMARU AAS 공개 저장소의 권리 문서는 네 개로 나뉘어 있고, 각 문서가 다루는 대상이 서로 다릅니다. LICENSE는 소스 코드에 적용되는 Apache-2.0 원문이고, NOTICE는 저작권 고지와 함께 이름·로고가 Apache-2.0의 허락 범위에 들어가지 않는다는 사실을 밝힙니다. TRADEMARKS는 회사 이름과 로고를 쓸 수 있는 범위를 정하고, THIRD_PARTY_NOTICES는 저장소 안에 파일째 들어 있는 다른 회사·개인의 소프트웨어와 자산을 라이선스별로 나열합니다. 도입 조직의 법무 검토는 이 네 문서를 함께 읽는 것으로 시작하면 됩니다.

검토 대상	적용 문서	도입 조직이 할 수 있는 일	지켜야 할 조건
소스 코드	LICENSE(Apache-2.0)	설치·사용·복제·수정·재배포, 상업적 이용, 특허 사용 허락	라이선스 사본 첨부, 바꾼 파일에 변경 사실 표시, NOTICE 고지 유지
이름·로고·마스코트	NOTICE·TRADEMARKS	프로젝트를 가리키거나 「기반으로 만들었다」고 밝히기, 바꾸지 않은 사본의 내부 사용	수정 배포본·서비스에서는 로고·마스코트를 빼고 제품명에 회사 이름을 쓰지 않음
제3자 구성요소	THIRD_PARTY_NOTICES	각 구성요소의 라이선스 범위 안에서 사용	구성요소별 저작권 고지·출처 표기 유지

Apache-2.0은 OSI(Open Source Initiative, 오픈소스 라이선스 인증 단체)가 승인한 허용형 라이선스입니다. 제2조는 저작권 사용 허락을 영구적이고 취소할 수 없는 형태로 주고, 제3조는 기여자가 가진 특허의 사용 허락까지 함께 줍니다[S14]. 도입 조직은 내려받은 코드를 사내 서버에 설치해 쓰고, 업무에 맞게 고치고, 고친 결과를 계열사나 고객에게 다시 배포할 수 있습니다. 다만 제4조는 재배포할 때 라이선스 사본을 함께 주고, 바꾼 파일에는 바꿨다는 표시를 남기고, NOTICE 파일의 고지를 유지하라고 요구합니다. 제7조와 제8조는 코드를 있는 그대로 제공하며 보증과 손해 책임을 지지 않는다고 정하므로, 운영 책임이 누구에게 있는지는 11.4의 책임 구분으로 따로 정해야 합니다.

11.1.1 수정 배포본에 쓸 수 없는 회사 이름이 들어간 제품명

상표 정책이 다루는 대상

Apache-2.0 제6조는 라이선스가 상표·상호·로고의 사용 허락을 포함하지 않는다고 명시합니다. OPENMARU AAS 저장소의 TRADEMARKS 문서는 이 조항에 따라 회사 이름 OPENMARU와 회사의 다른 제품 브랜드 이름, 저장소에 든 로고 파일 21개, 채팅 화면의 마스코트 이미지를 상표 정책 대상으로 정하고 있습니다. 이 때문에 코드를 자유롭게 고칠 권리와 이름을 쓸 권리는 처음부터 분리돼 있고, 법무 검토도 두 갈래로 나눠 진행해야 합니다. 소스 코드를 한 줄도 고치지 않고 사내에서 쓰는 경우에는 로고와 마스코트가 그대로 보여도 정책 위반이 아닙니다.

허용되는 사용과 지켜야 할 조건

TRADEMARKS 문서가 허용하는 사용은 두 가지입니다. 하나는 이 프로젝트를 이름으로 가리키거나 자기 결과물이 이 프로젝트를 기반으로 만들어졌다고 밝히는 일이고, 다른 하나는 로고와 마스코트가 들어 있는 바꾸지 않은 사본을 평가·개발·내부 용도로 실행하는 일입니다. 반면 수정한 판을 배포하거나 서비스로 제공할 때는 로고와 마스코트를 빼거나 바꿔야 하고, 제품 이름에 회사 이름과 회사 제품 브랜드를 쓸 수 없습니다. 그리고 어느 경우든 회사가 그 제품을 지원하거나 승인한 것처럼 보이게 이름이나 로고를 쓰면 안 됩니다[S14].

이 구분은 도입 조직이 공개 코드를 고쳐 자기 이름으로 사내 배포하거나 고객에게 납품하는 경우에 중요합니다. 제품 이름과 화면의 로고만 조직 것으로 바꾸면 코드와 기능은 그대로 쓸 수 있습니다. 반면 회사 이름을 넣은 제품명을 계속 쓰려면 공급사와 별도로 합의해야 하며, 문의는 공개 저장소의 이슈로 받습니다[S14].

11.1.2 3D 오피스 MIT 코드·OFL 글꼴·CC0 가구의 제3자 고지

저장소에 파일째 들어 있는 제3자 구성요소

THIRD_PARTY_NOTICES는 소스나 파일째 저장소에 들어 있는 구성요소만 다루고, npm·pip 패키지 의존성은 각 패키지의 라이선스를 따른다고 밝힙니다. 화면에서 에이전트들이 함께 일하는 3D 오피스는 MIT 라이선스 코드를 가져와 OPENMARU AAS가 관리하는 형태이고, 글꼴과 3D 모델, 배경음악은 각자의 공개 라이선스를 따릅니다. 법무 검토에서 확인할 대상은 아래 표의 여섯 묶음입니다.

구성요소	쓰는 곳	라이선스	재배포 때 남길 것
3D 오피스 장면 코드	오피스 화면	MIT	저작권 고지와 라이선스 원문
Noto Sans KR·JetBrains Mono 글꼴	화면 본문·고정폭 글자·3D 이름표	SIL Open Font License 1.1	라이선스 원문(글꼴 폴더의 OFL 파일)
사무실 가구·화분 3D 모델 28개	3D 오피스	CC0 1.0	의무 없음(출처 표기 선택)
부서 현수막 애드벌룬 3D 모델	3D 오피스	CC BY 3.0	제작자와 출처 표기
오피스 배경음악 5곡	오피스 배경음(기본 꺼짐)	CC0 1.0 4곡, CC BY 4.0 1곡	CC BY 4.0 곡의 작곡자·라이선스 표기
PDF·한글 문서 보기용 부가 파일	파일 뷰어	BSD 계열·MIT·Apache-2.0, 일부 글꼴은 GPLv2 + 글꼴 예외	각 폴더의 라이선스 원문

제3자 고지에서 확인할 의무와 폐쇄망 동작

표에서 출처 표기가 의무인 구성요소는 CC BY 계열 두 건이고, 나머지는 고지 파일을 그대로 두면 조건을 지킬 수 있습니다. 글꼴과 3D 모델을 저장소 안에 넣어 둔 이유는 폐쇄망에서 화면이 외부 서버에 요청하지 않게 하려는 것이며, 빌드할 때 네트워크를 막은 브라우저로 3D 오피스를 열어 외부 요청이 0건인지 검사합니다[S14]. 다만 공개 저장소 밖에서 따로 공급받는 구성요소가 있다면 그 목록과 라이선스는 이 고지에 들어 있지 않으므로, 조직 운영본을 받을 때 별도 목록을 요청해 같은 방식으로 검토하는 편이 안전합니다.

11.2 OSI 승인 라이선스 여부가 가르는 공급사 종속과 도입 위험

에이전트 빌더와 채팅 화면 분야에서 널리 쓰이는 「오픈소스」 도구 가운데 상당수는 OSI 승인 라이선스가 아닙니다. 소스 코드는 공개돼 있어도 상업적 재배포, 브랜드를 바꾼 납품, 다중 고객 서비스 제공을 막거나 별도 계약을 요구하는 조건이 붙어 있어서, 도입 조직이 자기 제품에 넣거나 계열사에 다시 나눠 주려 할 때 제약이 생깁니다. 라이선스 유형은 기술 검토보다 먼저 확인해야 하는 공급사 종속 위험의 문제입니다. OPENMARU AAS의 코드는 OSI 승인 허용형인 Apache-2.0 이므로 이 위험이 작은 편에 속합니다.

11.2.1 라이선스 유형별 사용·수정·재배포 제약 판단표

네 가지 라이선스 유형의 제약 비교

라이선스 유형은 내부 사용 허용 여부보다 수정본을 다시 내보낼 때의 조건에서 크게 갈립니다. 아래 표는 제품 이름 없이 유형만으로 비교한 것이며, 실제 도입 대상의 라이선스 원문으로 각 칸을 다시 확인해야 합니다.

유형	예	내부 설치·수정	브랜드를 바꾼 재배포·납품	여러 고객 대상 서비스 제공	공급사 사정이 바뀔 때
OSI 승인 허용형	Apache-2.0, MIT	가능	가능(고지 유지, 상표는 별도)	가능	받은 판을 계속 쓰고 고칠 수 있음
OSI 승인 상호주의형	GPL·AGPL 계열	가능	가능하지만 수정본 소스 공개 의무	AGPL은 서비스 이용자에게도 소스 공개 의무	받은 판을 계속 쓸 수 있음
소스 공개 제한형	상업적 이용·서비스 제공을 제한하는 공개 라이선스	대체로 가능	제한되거나 별도 계약 필요	제한되거나 별도 계약 필요	다음 판의 조건이 바뀔 수 있음
상용 라이선스	구독·영구 사용권	계약 범위 안에서만	원칙적으로 불가	계약 범위 안에서만	계약 종료 때 사용 중단 위험

결재 문서에 남길 라이선스 판단 문장

표에서 의사결정권자가 볼 칸은 「공급사 사정이 바뀔 때」입니다. 공급사가 다음 판부터 라이선스를 제한형으로 바꾸더라도 OSI 승인 허용형으로 받은 판은 그 조건에 묶이지 않고, Apache-2.0은 이미 받은 판에 대한 사용 허락을 취소할 수 없다고 정합니다[S14]. 반면 소스 공개 제한형과 상용 라이선스는 조건 변경이나 계약 종료가

곧바로 도입 조직의 운영 위험이 됩니다. 결재 문서에는 「코드 라이선스는 OSI 승인 허용형이며, 상표와 제3자 구성요소는 별도로 검토했다」는 문장 하나를 남기는 것으로 충분합니다.

11.2.2 공개 코드를 직접 설치·수정할 때 줄어드는 공급사 종속

공급사 종속 위험의 두 경우 비교

공급사 종속은 공급사가 사업 방향을 바꾸거나 지원을 끊었을 때 도입 조직이 계속 운영할 수 있는가로 판단합니다. OPENMARU AAS는 소스 코드가 공개돼 있고, 에이전트 데이터는 조직이 관리하는 저장소 볼륨에 SQLite 데이터베이스와 파일로 남습니다. 공급사 사정과 무관하게 조직이 코드를 유지할 수 있으며, 아래 두 경우의 차이는 결재 항목으로 바로 옮길 수 있습니다.

경우	공급사가 방향을 바꾸거나 지원을 끊을 때	도입 조직이 할 수 있는 일
공개 코드를 직접 설치·수정	받은 판의 사용 권리가 그대로 남음	사내 개발자나 협력사가 코드를 고쳐 계속 운영하고, 데이터 볼륨을 백업·이전
공급사가 운영하는 서비스형 빌더	서비스 조건 변경·종료에 따름	에이전트 정의와 기록을 내보낼 수 있는 범위 안에서만 이전

공개 코드로도 남는 종속 요인

다만 코드 공개가 모든 종속을 없애지는 않습니다. OPENMARU AAS가 지금 연결하는 에이전트 플랫폼은 Hermes Agent 하나이므로 런타임 쪽 종속은 남아 있고, 이 한계는 12장의 발전 단계에서 다룹니다. 모델 쪽 종속은 작습니다. OpenAI 호환 주소 하나만 있으면 사내 vLLM·LiteLLM·Ollama 가운데 어느 것이든 연결할 수 있고, 도구 호출과 64,000 토큰 이상의 문맥을 지원하는 모델이면 됩니다[S14]. 도입 조직은 모델은 바꿔 가며 쓰고, 런타임은 공개 오픈소스인 Hermes Agent를 함께 운영한다는 전제로 종속 위험을 평가하면 됩니다.

11.3 Docker Compose 공개본과 비공개 쿠버네티스 구성의 기능 경계

「코드를 직접 설치하고 고칠 수 있다」는 말은 공개된 범위 안에서만 맞는 말입니다. OPENMARU AAS의 공개본은 Docker Compose로 AI 에이전트 작업공간 하나를 실행하는 구성이고, 사용자마다 작업공간을 따로 두는 쿠버네티스 구성에 필요한 오퍼레이터와 Helm 차트는 공개하지 않았습니다[S14]. 그래서 공개본만으로는 일부 기능이 꺼지며, IT 담당자는 도입 형태를 정하기 전에 아래 경계표로 필요한 기능이 어느 쪽에 있는지 확인해야 합니다.

기능	Docker Compose 공개본	쿠버네티스 구성(조직 운영본)
자체 계정 로그인, SSO(OIDC) 연결	된다	된다
대시보드·실행 기록·스킬·스킬 연결도·파일 보기	된다	된다
채팅·3D 오피스·칸반·회의	된다	된다
예약 작업·메신저 연결	된다	된다

기능	Docker Compose 공개본	쿠버네티스 구성(조직 운영본)
프로파일 만들기·성격(SOUL)·비밀값·MCP 서버 연결	된다	된다
스킬 허브 설치	꺼짐	된다
AI 에이전트 작업공간 만들기·지우기	꺼짐	된다
작업 지침(AGENTS.md)	꺼짐	된다
사용자마다 별도 Hermes 작업공간	작업공간 하나	사용자마다 하나

11.3.1 공개본에서 꺼지는 스킬 허브 설치·작업공간 만들기·작업 지침

공개본에서 꺼지는 세 기능과 이유

공개본 README는 Docker Compose에서 꺼지는 기능 세 가지와 그 이유를 밝히고, 화면에도 같은 이유를 보여 줍니다. 스킬 허브 설치와 AI 에이전트 작업공간 만들기·지우기는 쿠버네티스가 있어야 동작하므로 공개본에서는 꺼집니다. 그렇더라도 공개본에서 스킬을 직접 만들어 쓰는 일은 되고, 작업공간 하나 안에 프로파일을 여러 개 만들어 에이전트를 나눠 쓸 수 있습니다. 세 번째로 꺼지는 작업 지침(AGENTS.md)은 공개 Hermes 이미지에 그 기능의 API가 없어서 쓸 수 없는 경우입니다.

공개 Hermes 이미지에는 OPENMARU AAS 운영에 쓰는 일부 수정 사항이 빠져 있습니다. 이 때문에 MCP OAuth 연결 상태가 「확인 전」으로 보이고, 에이전트가 자기를 Hermes Agent로 소개하며, 한국어로 도구를 찾을 때 정확도가 조금 낮습니다[S14]. 따라서 공개본은 한 부서가 작업공간 하나로 첫 업무를 시범 도입해 보는 용도에 맞고, 여러 부서가 사내 허브에서 스킬을 받아 쓰는 단계에서는 쿠버네티스 구성을 검토해야 합니다.

공개본 설치 조건과 compose 파일 세 종류

공개본의 준비물은 Docker Compose v2, 약 4.5GB의 Hermes 이미지를 받을 디스크 공간, 그리고 문맥이 64,000 토큰 이상인 OpenAI 호환 LLM 주소 하나입니다. 저장소에는 목적이 다른 compose 파일이 세 개 있습니다. docker-compose.yml은 Hermes, 읽기 사이드카, AAS를 하나씩 실행하는 새 설치용이고, docker-compose.attach.yml은 이미 운영 중인 Hermes에 읽기 사이드카와 AAS만 붙이는 경우에 씁니다. 마지막 docker-compose.multi.yml은 Hermes 세 개와 AAS 하나를 띄워 사용자마다 Hermes를 두는 구조를 개발 환경에서 시험하는 용도이므로 운영 설치에는 쓰지 않습니다.

폐쇄망에서는 이미지 주소를 조직의 내부 레지스트리로 바꿔 지정할 수 있으므로, 외부 인터넷이 막힌 망에서도 공개본을 설치할 수 있습니다[S14].

11.3.2 공개하지 않은 쿠버네티스 오퍼레이터와 Helm 차트

쿠버네티스 구성에만 있는 구성요소

쿠버네티스 모드는 사용자마다 Hermes를 하나씩 두는 구성이며, 이 모드를 쓰려면 Hermes 에이전트를 사용자 정의 리소스(HermesAgent)로 관리하는 쿠버네티스 오퍼레이터가 필요합니다. 오퍼레이터는 쿠버네티스에서 특정 애플리케이션의 설치·복구·확장을 자동으로 처리하는 관리 프로그램이고, Helm 차트는 그 구성을 한 번에 설치하는 패키지입니다. 이 두 가지는 공개 저장소에 없습니다. 관리자 매뉴얼이 설명하는 작업공간 개수 한도, 재시작, 이미지 배포, 스킬 허브의 공용·제한 구역과 게시 검토는 이 구성 위에서 동작합니다[S13].

쿠버네티스 구성 도입 때 확인할 항목

조직 운영본을 검토할 때 IT 담당자가 공급사에 먼저 확인할 것은 오퍼레이터와 Helm 차트를 어떤 조건과 형태로 받는지, 소스 코드도 함께 받는지입니다. 이어서 오퍼레이터·차트에 포함된 구성요소의 라이선스 목록을 11.1.2와 같은 형식으로 받아 두면 법무 검토를 공개본과 같은 방식으로 끝낼 수 있습니다. 운영 쪽에서는 지원하는 쿠버네티스 버전, 폐쇄망 내부 레지스트리 사용 방법, 사용자당 작업공간 개수 한도와 자원 요구량, 오래 쓰지 않은 작업공간의 처리 방식을 확인합니다. 마지막으로 공개본에서 쿠버네티스 구성으로 옮길 때 기존 작업공간 데이터를 이전하는 절차가 있는지 물어 두어야 시범 도입에서 만든 스킬을 버리지 않고 확산 단계로 가져갈 수 있습니다.

이 항목에 답을 받으면 「공개본으로 시범 도입하고, 부서 확산 단계에서 쿠버네티스 구성으로 옮긴다」는 도입 순서를 결재 문서에 그대로 쓸 수 있습니다.

11.4 「오픈소스면 지원은 누가 하나」라는 질문과 공개 코어·조직 스킬·사내 허브의 책임 구분

오픈소스 도입을 검토하는 조직이 가장 먼저 묻는 질문은 문제가 생겼을 때 누가 고치느냐입니다. Apache-2.0은 코드를 보증 없이 제공한다고 정하면서도, 제9조에서 재배포하는 쪽이 별도 대가를 받고 지원이나 보증을 제공하는 것을 허용합니다[S14]. 그래서 지원 책임은 라이선스 한 줄이 아니라 OPENMARU AAS를 구성하는 세 층마다 따로 정해야 하며, 공개 코어·조직 스킬·사내 허브로 나누면 책임 경계가 분명해집니다.

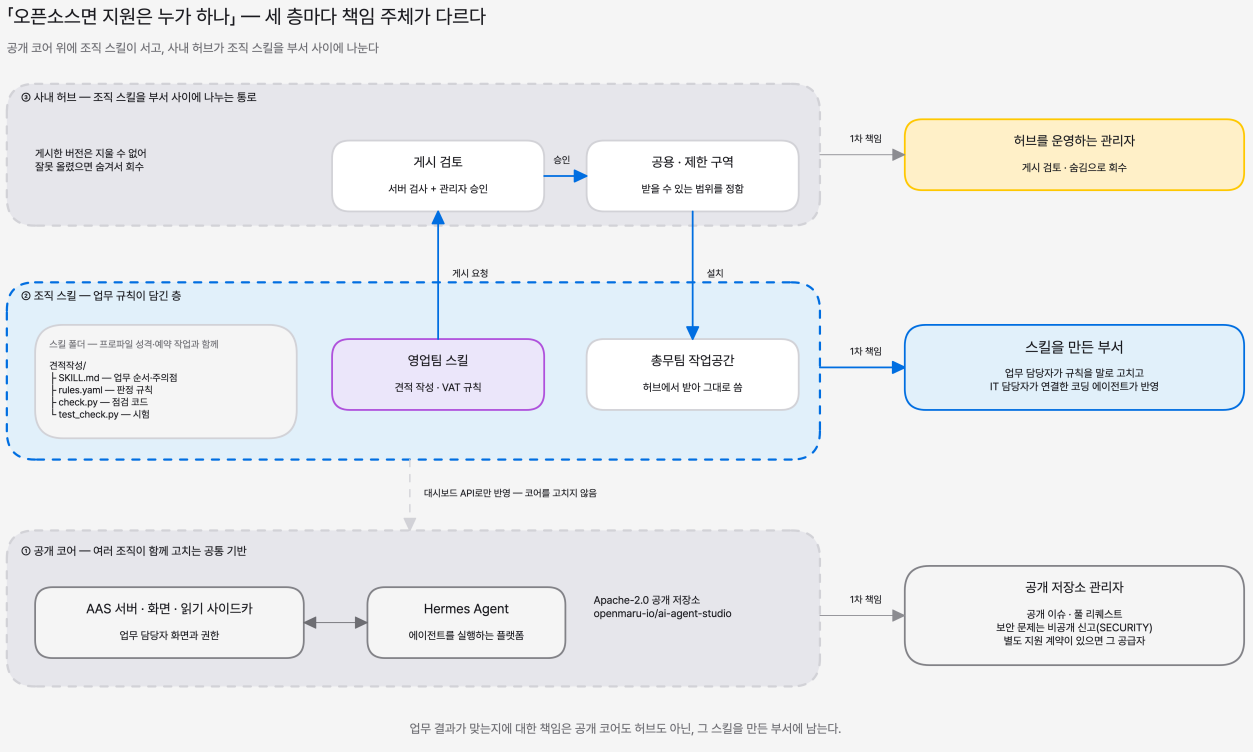


그림. 공개 코어·조직 스킬·사내 허브의 3층 구조와 층별 책임 주체 — 업무 결과의 정확성 책임은 스킬을 만든 부서에 남는다

층	무엇이 들어 있나	1차 책임	결함·변경 요청 경로
공개 코어	AAS 서버·화면·읽기 사이드카 코드, 함께 쓰는	공개 저장소 관리자, 별도 지원 계약이 있으면 그 공	공개 이슈·풀 리퀘스트, 보안 문제는 비공개 신고

층	무엇이 들어 있나	1차 책임	결함·변경 요청 경로
	Hermes Agent	급자	
조직 스킬	부서가 만든 스킬·스크립트·프로파일 성격·예약 작업	스킬을 만든 부서의 업무 담당자와 IT 담당자	업무 담당자가 규칙을 말로 고치고 코딩 에이전트가 반영
사내 허브	공용·제한 구역에 게시된 스킬, 게시 검토 기록	허브를 운영하는 관리자	게시 검토와 회수

이 구분에서 의사결정권자가 기억할 사실은 업무 규칙이 담긴 층이 조직 스킬이라는 점입니다. 공개 코어는 여러 조직이 함께 고치는 공통 기반이고, 사내 허브는 조직 스킬을 부서 사이에 나누는 통로이므로, 업무 결과의 정확성에 대한 책임은 결국 조직 스킬을 만든 부서에 남습니다. 보안 문제는 공개 이슈로 올리지 않고 SECURITY 문서가 정한 비공개 경로로 신고하게 돼 있어, 취약점 정보가 먼저 공개되는 일을 막습니다[S14].

11.4.1 에이전트를 만드는 개발자에게 열린 스킬·MCP 연동의 확장 지점

스킬과 MCP 연결로 넓히는 확장 방법

에이전트를 만드는 개발자가 하는 확장의 대부분은 OPENMARU AAS 코드 수정 없이 처리됩니다. AAS는 바꾸거나 대화할 때 Hermes 대시보드 API를 부르고, Hermes가 변경을 검사해 저장하며, AAS 자신은 Hermes 데이터 폴더에 직접 쓰지 않습니다. 개발자가 만든 스킬과 연결 설정은 Hermes 작업공간에 남고, AAS를 새 판으로 올려도 그대로 유지됩니다. 사용자 매뉴얼에 나오는 확장 방법은 다음과 같습니다[S12].

- 스킬 만들기: 화면에서 스킬 문서와 스크립트를 직접 만들거나, 만든 스킬을 사내 허브에 게시합니다.
- MCP 서버 연결: MCP(Model Context Protocol, 에이전트가 외부 도구를 부르는 표준 규약) 서버를 프로파일마다 연결해 사내 시스템 API를 도구로 씁니다.
- 프로파일 성격과 모델: 성격 문서(SOUL)와 사용할 모델을 프로파일마다 정합니다.
- 예약 작업과 메신저 연결: 정해진 시각에 실행할 작업과 결과를 보낼 메신저를 지정합니다.
- Claude Code 연결: 사용자 PC의 Claude Code를 작업공간에 연결해 스킬을 만들어 올리고 실제 AI 에이전트로 시험합니다.
- 가져오기와 내보내기: 프로파일과 스킬을 파일로 옮겨 다른 작업공간에 설치합니다.

확장할 때 구분할 만드는 단계와 일하는 단계

이 확장 방법은 만드는 단계와 일하는 단계에 따라 쓰는 모델이 다릅니다. 만드는 단계에서는 업무 담당자가 말로 정한 규칙을 코딩 에이전트가 코드로 옮기는데, 지금은 외부 코딩 에이전트인 Claude Code를 연결해 쓰고 폐쇄망이면 사내 코딩 모델로 바꿀 수 있습니다. 반면 일하는 단계에서는 사내 sLLM 에이전트가 완성된 스킬을 매일 실행하며, 모든 모델 호출이 조직의 모델 게이트웨이를 거치므로 고객·금액 자료가 조직 밖 LLM으로 나가지 않습니다[S13]. 개발자가 코드를 고쳐야 하는 경우는 공개 코어 자체의 결함이나 화면 기능 추가로 좁혀지고, 그 변경은 11.4.2의 수정 배포 조건을 따릅니다.

11.4.2 공개 저장소 openmaru-io/ai-agent-studio와 상표 정책 안의 수정 배포

공개 저장소와 기여 방식

공개 저장소 주소는 github.com/openmaru-io/ai-agent-studio 이며, 도입 조직이 찾은 결함이나 개선 제안은 공개 이슈와 풀 리퀘스트로 보냅니다. 풀 리퀘스트를 열려면 모든 커밋에 DCO(Developer Certificate of Origin, 기여자가 그 코드를 제출할 권리가 있음을 밝히는 서명) 줄이 있어야 합니다[S14]. 이 방식이면 조직이 고친 내용이 공개 코어에 합쳐져 다음 판부터 조직이 따로 유지할 수정분이 줄어듭니다. 반대로 조직 고유 업무에 묶인 수정은 공개하지 않고 사내 판으로 유지할 수 있으며, Apache-2.0은 수정본의 소스 공개를 요구하지 않습니다.

수정 배포 전에 확인할 점검 항목

고친 판을 계열사나 고객에게 배포하거나 서비스로 제공하기 전에는 다음 항목을 확인합니다[S14].

점검 항목	근거 문서	확인 방법
Apache-2.0 라이선스 사본을 배포본에 넣었는가	LICENSE	배포 묶음에 LICENSE 파일 포함
바꾼 파일에 변경 사실을 표시했는가	LICENSE 제4조	파일 머리말이나 변경 목록
NOTICE의 저작권 고지를 유지했는가	NOTICE	배포 묶음에 NOTICE 파일 포함
제3자 구성요소의 고지와 출처 표기를 유지했는가	THIRD_PARTY_NOTICES	CC BY 구성요소의 표기 확인
로고 파일과 마스코트를 빼거나 바꿨는가	TRADEMARKS	화면·문서의 로고 교체
제품 이름에 회사 이름과 회사 제품 브랜드가 없는가	TRADEMARKS	제품명·도메인·문서 제목 확인
공급사가 지원하거나 승인한 것처럼 표기하지 않았는가	TRADEMARKS	소개 문구 검토

바꾸지 않은 사본을 사내에서만 쓰는 조직은 이 표의 앞 세 항목도 따로 처리할 필요가 없습니다. 수정 배포를 하지 않는 시범 도입 단계에서는 라이선스 검토가 사실상 끝나고, 수정본을 밖으로 내보내는 시점에 이 점검표를 한 번 적용하면 됩니다.

12장: 단계별 발전 방향과 도입 순서

라이선스 검토까지 마쳤다면 남는 질문은 제품이 지금 어디까지 왔고 우리 조직이 어떤 순서로 시작하느냐입니다. OPENMARU AAS는 「내 에이전트가 보인다」에서 「다른 에이전트 플랫폼에 붙는다」까지 여덟 단계로 발전 방향을 정해 두고 있으며, 2026년 11월 현재 앞의 네 단계를 제공하고 다음 두 단계를 일부 제공합니다. 도입을 검토하는 조직에는 제품이 어디까지 왔는지와 함께, 자기 조직이 어떤 순서로 시작해야 실패 위험을 줄일 수 있는지가 중요합니다. 2장에서 본 것처럼 Gartner는 2027년 말까지 에이전틱 AI 프로젝트의 40% 이상이 비용 상승, 불분명한 비즈니스 가치, 부족한 위험 통제 때문에 취소될 것으로 전망했습니다[S4]. 제품의 현재 위치와 남은 한계, 도입 4단계와 단계별 중단 기준, 첫 업무 선정 기준을 차례로 확인하면, 12.4의 도입 결재 점검표로 의사결정권자가 한 장에서 결재할 수 있습니다.

12.1 발전 8단계에서 본 OPENMARU AAS의 현재 위치와 다음 순서

OPENMARU AAS의 발전 8단계는 한 사람이 자기 에이전트를 보고 고치는 데서 시작해, 사람마다 자기 에이전트를 받고, 여러 에이전트와 사람이 함께 일하고, 조직 단위로 운영하는 순서로 넓어집니다. 단계마다 상태를 「제공」, 「일부 제공」, 「착수 전」으로 나눠 적으면 제품 성숙도를 과장 없이 판단할 수 있습니다. 로드맵은 순서만 정해 두었고 일정은 약속하지 않으므로, 아래 표의 「착수 전」 단계를 도입 조건으로 삼는 조직은 그 기능이 나온 뒤에 다시 판단해야 합니다.

단계	내용	상태(2026년 11월)
1. 내 에이전트가 보인다	예약 작업의 실패·멈춤, 스킬 호출 흐름, 활동, 메신저 채널, 모델, 자격증명을 한 화면에서 본다	제공
2. 그 자리에서 고친다	예약 작업·스킬 문서·프로파일·비밀값을 화면에서 고친다	제공(스크립트는 보기만)
3. 콘솔 안에서 일을 시킨다	에이전트를 골라 채팅하고 승인·되묻기에 답하며 지난 대화를 잇는다	제공
4. 사람마다 자기 에이전트	표준 로그인, 사용자마다 에이전트 추가·소유권 이전·다시 시작	제공
5. 에이전트끼리, 사람과 함께	3D 오피스·칸반·회의실에서 여러 에이전트가 함께 일한다	일부 제공(상시 방만, 만들고 지우는 방은 없음)
6. 받아서 쓴다	사내 스킬 허브에서 스킬을 설치하고 게시한다	일부 제공(공용·제한 구역과 게시 검토, 결제 없음)
7. 조직에 넣는다	팀·회사 공용 에이전트, 관리 주체, 감사	착수 전(권한 4종·부서·변경 기록이 기반 기능)
8. 다른 에이전트 플랫폼에 붙는다	런타임 어댑터	착수 전(지금은 Hermes Agent 하나)

다음 순서는 세 가지로 정해져 있습니다. 먼저 필요할 때 만들고 끝나면 지우는 협업 방을 만들고, 이어서 권한 구분과 결제가 있는 사내 마켓을 운영하며, 그다음에 두 번째 에이전트 플랫폼을 연결합니다. 이 순서는 5·6·8단계의 「일부 제공」과 「착수 전」을 앞에서부터 채우는 방향이며, 7단계의 팀·회사 공용 에이전트는 조직 도입 사례가 생길 때 관리 주체를 함께 정하는 과제로 남겨 두었습니다.

7단계가 착수 전이라는 사실은 도입 범위를 정할 때 함께 고려해야 합니다. 관리자 매뉴얼에 있는 관리자·팀 관리자·작업공간 주인·일반 사용자의 네 가지 권한, 사용자와 부서 관리, 사용량과 변경 기록은 이미 제공되므로 한 부서의 시범 도입과 몇 개 부서로의 확산에는 부족함이 없습니다[S13]. 다만 여러 사람이 같은 에이전트 하나에 일을 시키는 팀 공용 에이전트는 아직 없으므로, 그런 업무는 사용자마다 자기 에이전트에 같은 스킬을 설치해 나눠 쓰는 방식으로 운영합니다.

12.1.1 사용자·관리자 매뉴얼과 코딩 에이전트 연결 등 최근 추가 기능

공개 문서와 공개 저장소의 추가

최근 가장 큰 변화는 제품을 처음 보는 조직이 스스로 평가할 수 있는 공개 자료가 생겼다는 점입니다. 사용자 매뉴얼은 시작하기부터 기록과 설정까지 7부 38쪽, 관리자 매뉴얼은 구성과 권한부터 연동 서비스까지 8부 31쪽으로 docs.openmaru.io에 공개돼 있습니다[S12][S13]. 또한 소스 코드가 Apache-2.0으로 공개되고 Docker Compose로 설치하는 공개본이 함께 나와, IT 담당자가 공급사를 거치지 않고 시험 설치를 해 볼 수 있습니다[S14]. 이 두 자료는 도입 4단계의 첫 두 단계에서 IT 담당자가 바로 쓸 수 있는 근거 문서입니다.

매뉴얼에 실린 최근 화면 기능

매뉴얼에 실린 기능 가운데 최근에 더해진 것은 업무 담당자와 관리자가 매일 쓰는 화면에 몰려 있습니다. Claude Code 연결은 사용자 PC의 Claude Code를 작업공간에 연결해 스킬을 만들어 올리고 실제 AI 에이전트로 시험하는 기능이며(사용자 매뉴얼 7.5), 관리자는 모든 사람의 연결을 한 화면에서 보고 강제로 끊을 수 있습니다(관리자 매뉴얼 7.4)[S12][S13]. 스킬 연결도는 스킬이 어디서 불리고 무엇을 부르는지 그림으로 보여 주고(사용자 매뉴얼 5.6), 문서 뷰어는 채팅에서 받은 PDF·Office·한글 문서를 내려받지 않고 화면 안에서 엽니다(사용자 매뉴얼 2.4). 말로 입력과 읽어 주기는 질문을 말로 넣고 답을 소리로 듣게 합니다(사용자 매뉴얼 2.5)[S12]. 관리자 쪽에서는 역할별로 미리 써 둔 성격(SOUL) 글인 역할 템플릿과, 정한 기간 동안 대상 사용자의 모든 화면 맨 위에 띠로 보이는 공지가 더해졌습니다(관리자 매뉴얼 5.1·5.2)[S13].

이 가운데 Claude Code 연결은 6장의 만드는 단계에서 코딩 에이전트가 만든 스킬을 실제 AI 에이전트로 시험하는 경로를 AAS 안에 마련한 기능입니다. 지금은 외부 코딩 에이전트를 쓰지만, 폐쇄망 조직은 같은 단계를 사내 코딩 모델로 바꿔 운영할 수 있습니다.

12.1.2 단일 플랫폼 연동·비용 통제 등 남은 한계와 보완 계획

도입 전에 알아야 할 남은 한계

8.4에서 비교한 부족한 영역은 도입 조직의 운영 조건으로 바꿔 읽어야 합니다. 아래 표는 한계마다 보완 방향과 그동안 쓸 수 있는 운영 방식을 함께 적은 것이며, 보완 시기는 정해져 있지 않습니다.

남은 한계	보완 방향	그동안의 운영 방식
연결되는 에이전트 플랫폼이 Hermes Agent 하나	8단계 런타임 어댑터	Hermes Agent를 공개 오픈소스 런타임으로 함께 운영

남은 한계	보완 방향	그동안의 운영 방식
비용 통제는 사용량 보기만 제공	사용량 한도와 통제 기능	모델 게이트웨이(LiteLLM 등)의 예산·속도 제한 사용
사내 마켓에 결제가 없음	권한 구분과 결제가 있는 마켓 운영	공용·제한 구역과 게시 검토로 공유 범위만 관리
만들고 지우는 협업 방이 없음	협업 방 생성·삭제와 일정 초대	상시 오피스·칸반·회의실 사용
스크립트는 화면에서 보기만 됨	화면 편집 범위 확대	업무 담당자는 규칙을 말로 고치고 코딩 에이전트가 반영
변경 기록은 AAS를 거친 변경만 남김	7단계 감사 기능	운영 변경을 AAS 화면으로만 하도록 권한 제한
개발 경험이 없는 부서의 업무 담당자가 만든 공개 사례가 적음	사례 확대	시범 도입에서 업무 담당자와 IT 담당자를 짝으로 배정

한계를 결재 조건으로 바꾸는 방법

표의 「그동안의 운영 방식」 열은 한계가 시범 도입을 막는지 판단하는 데 씁니다. 예를 들어 비용 통제 기능이 없더라도 모든 모델 호출이 조직의 모델 게이트웨이를 거치므로, 게이트웨이의 예산 한도로 시범 도입 기간의 비용을 묶어 둘 수 있습니다[S13]. 반면 두 번째 에이전트 플랫폼이 반드시 필요한 조직이라면 지금 도입하기보다 8단계가 제공될 때 다시 판단하는 편이 맞습니다.

12.2 AI 단계 진단·sLLM 준비·첫 업무 정의·부서 확산의 도입 4단계

OPENMARU AAS의 도입은 AI 단계 진단, 내부 sLLM 준비, 첫 업무 정의, 부서 확산의 네 단계로 나누며, 각 단계는 결재 산출물 하나로 끝납니다. 이렇게 나누는 이유는 생성형 AI 사업의 상당수가 PoC 뒤에 멈추기 때문입니다. Gartner는 2025년 말까지 생성형 AI 프로젝트의 최소 30%가 데이터 품질, 위험 통제, 비용, 불분명한 가치 문제로 PoC 이후 중단될 것으로 전망했습니다[S3]. 또한 에이전트를 만들 때는 가장 단순한 해법에서 시작해 필요할 때만 복잡도를 높이라는 권고가 있으며[S11], 네 단계도 그 원칙에 따라 한 부서의 한 업무에서 출발합니다.

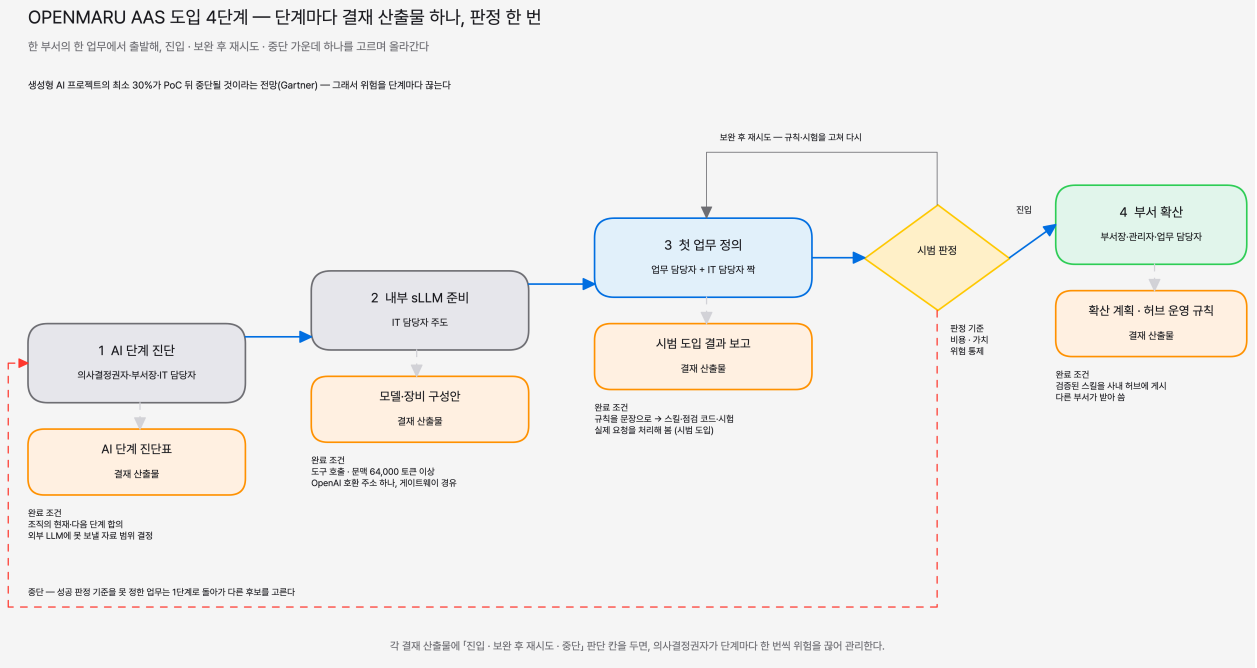


그림. AI 단계 진단 → 내부 sLLM 준비 → 첫 업무 정의(시범 도입) → 부서 확산의 4단계와 단계별 결재 산출물, 진입·재시도·중단 판단

단계	주 참여자	완료 조건	결재 산출물
1. AI 단계 진단	의사결정권자, 업무 부서장, IT 담당자	3장의 다섯 단계 가운데 조직의 현재 단계와 다음 단계를 합의하고, 외부 LLM에 보낼 수 없는 자료의 범위를 정함	AI 단계 진단표
2. 내부 sLLM 준비	IT 담당자	도구 호출과 64,000 토큰 이상의 문맥을 지원하는 모델을 OpenAI 호환 주소 하나로 제공하고, 모든 호출이 모델 게이트웨이를 거치게 함	모델·장비 구성안
3. 첫 업무 정의(시범 도입)	업무 담당자, IT 담당자	업무 규칙을 문장으로 정하고 스킬·점검 코드·시험을 만들어 실제 요청을 처리함	시범 도입 결과 보고
4. 부서 확산	부서장, 관리자, 업무 담당자	검증된 스킬을 사내 허브에 게시하고 다른 부서가 받아 씀	확산 계획과 허브 운영 규칙

2단계의 기술 조건은 공개 저장소와 관리자 매뉴얼에 그대로 나와 있습니다. Hermes Agent는 도구를 쓰려면 문맥이 64,000 토큰 이상이어야 하고[S14], OPENMARU AAS의 에이전트는 LiteLLM 같은 모델 게이트웨이를 거쳐 모델을 부릅니다[S13]. 따라서 IT 담당자가 이 두 조건을 갖춘 주소 하나를 준비하면 3단계로 넘어갈 수 있습니다.

12.2.1 단계마다 달라지는 업무 담당자와 IT 담당자의 몫

단계별 역할 분담

5.2에서 나눈 책임은 도입 단계마다 비중이 달라집니다. AI 단계 진단에서는 업무 담당자가 반복 업무와 쓰는 자료를 설명하고, IT 담당자가 지금 쓰는 AI 도구와 자료가 조직 밖으로 나가는 경로를 파악하며, 의사결정권자가 다음 단계 목표와 외부 LLM 사용 범위를 정합니다. 내부 sLLM 준비 단계는 IT 담당자가 주도해 모델 서빙, 모델 게이트웨이, SSO 연결, 설치 형태를 정하고, 이 동안 업무 담당자는 시범 업무 후보를 미리 고르며 의사결정권자는 장비 도입 여부를 결정합니다.

첫 업무 정의 단계부터는 업무 담당자가 중심이 됩니다. 업무 담당자는 규칙을 문장으로 정하고 결과를 검수하며, IT 담당자는 코딩 에이전트 연결, MCP를 통한 사내 시스템 연동, 권한 설정을 맡고, 의사결정권자는 성공 판정 기준을 승인합니다. 부서 확산 단계에서는 업무 담당자가 스킬을 다른 부서에 설명하며 규칙을 계속 고치고, 관리자가 허브 구역과 게시 검토, 작업공간 한도를 운영하며, 의사결정권자는 확산 대상 부서와 예산을 승인합니다.

업무 담당자와 IT 담당자의 짝 배정

3단계에서 업무 담당자와 IT 담당자를 한 쌍으로 묶는 이유는 만드는 단계의 두 가지 일이 서로 다른 사람의 몫이기 때문입니다. 업무 담당자는 견적의 VAT 처리나 조달 등록가 계산처럼 업무 규칙을 문장으로 정하고 결과가 맞는지 판정합니다. 반면 코딩 에이전트 연결, 사내 시스템을 도구로 부르는 MCP 연동, 사용자 권한은 IT 담당자의 일입니다. 관리자 매뉴얼은 사용자 권한을 관리자, 팀 관리자, 작업공간 주인, 일반 사용자의 네 가지로 나누므로, 시범 도입에서는 업무 담당자를 자기 작업공간의 주인으로 두고 IT 담당자를 관리자나 그 부서의 팀 관리자로 지정하는 구성이 출발점이 됩니다[S13].

12.2.2 다음 단계 진입 조건과 중단 판단 기준

비용·가치·위험 통제의 세 기준

Gartner가 꼽은 에이전틱 AI 프로젝트 취소 원인은 비용 상승, 불분명한 비즈니스 가치, 부족한 위험 통제의 세 가지입니다[S4]. 이 세 원인을 단계마다 진입 조건과 중단 기준으로 바꿔 두면, 실패를 늦게 발견해 예산을 소진하는 경우를 막고 작은 단계에서 멈출 수 있습니다.

기준	다음 단계로 가는 조건	중단하거나 되돌아가는 조건
비용	모델 호출이 게이트웨이를 거치고 시범 기간 예산 한도 안에서 처리됨	장비·인력 비용이 업무 한 건의 절감 효과를 계속 넘음
가치	첫 업무의 처리 시간·검수 수정량이 시범 전보다 줄었고 업무 담당자가 계속 쓰기로 함	성공 판정 기준을 정하지 못했거나, 결과를 매번 사람이 다시 만듦
위험 통제	고객·금액 자료가 조직 밖 LLM으로 나가지 않고, 잘못된 입력이 점검 단계에서 멈춤	자료 반출 경로가 남아 있거나, 점검 없이 외부 시스템에 기록이 써짐

중단 판단을 결재 문서에 넣는 방식

중단 기준을 미리 결재받아 두면 시범 도입이 기대에 못 미쳤을 때도 미리 정한 절차에 따라 다음 판단을 내릴 수 있습니다. 예를 들어 3단계에서 성공 판정 기준을 정하지 못했다면 그 업무는 규칙이 아직 문장으로 정리되지 않은 업무이므로, 1단계로 돌아가 다른 후보 업무를 고르는 것이 맞습니다. 이처럼 각 단계의 결재 산출물에 「진입」, 「보완 후 재시도」, 「중단」 세 가지 판단 칸을 두면 의사결정권자가 단계마다 한 번씩 위험을 끊어 관리할 수 있습니다.

시범 운영 중에는 사람이 처리한 결과로 판단

시범 운영 중의 판단은 화면에 드러난 문제를 사람이 실제로 처리했는지로 내립니다. 화면은 원인을 드러낼 뿐이고, 결과를 받을 곳을 지정하거나 만료된 키·토큰을 바꾸는 것은 담당자이기 때문입니다. 판단은 세 갈래로 정리할 수 있습니다. 첫째, 처음에 정한 성공 판정 기준(처리 시간, 결과 정확도, 사람 확인 단계에서 걸러진 건수)을 충족하면 시범 운영을 계속합니다. 둘째, 알림 없는 작업과 만료 임박 키·토큰이 0개로 유지되고 같은 업무를 원하는 다른 부서가 있으면 부서 확산 단계로 넘어갑니다. 셋째, 화면이 같은 문제를 거듭 드러내는데도 처리할 사람이 정해지지 않으면 확산을 멈추고 담당 체계부터 다시 정합니다.

12.3 시범 도입 대상이 될 첫 업무의 선정 기준 — 말로 정할 수 있는 규칙·반복 빈도·조직 안에 둘 자료

시범 도입의 성패는 첫 업무를 무엇으로 고르느냐에 크게 좌우됩니다. 6.4의 판별표를 도입 조직 입장에서 다시 쓰면 첫 업무는 세 가지 조건을 함께 갖춰야 합니다. 업무 규칙을 업무 담당자가 문장으로 말할 수 있어야 하고, 같은 일이 자주 반복돼야 하며, 다루는 자료가 조직 밖으로 나가면 안 되는 업무여야 합니다. 세 조건을 모두 갖춘 업무는 내부 sLLM 에이전트로 처리할 이유와 효과가 함께 분명해지므로, 아래 점검표에서 「예」가 가장 많은 업무 하나를 시범 도입 대상으로 고릅니다.

점검 질문	「예」일 때 의미	「아니오」일 때
업무 담당자가 규칙을 문장 열 줄 남짓으로 말할 수 있는가	규칙을 코드로 옮겨 판단을 코드가 맡을 수 있음	규칙이 사람의 감에 있으므로 후보에서 뺌
같은 업무가 매주 여러 번 반복되는가	처리 시간 절감이 바로 보임	효과가 작아 확산 근거가 약함
고객·금액·개인 자료처럼 밖으로 보내면 안 되는 자료를 다루는가	내부 sLLM 에이전트가 필요한 이유가 분명함	외부 서비스로도 처리할 수 있어 우선순위가 낮음
결과가 맞는지 판정할 기준이 있는가	성공 판정과 점검 코드를 만들 수 있음	시범 도입 결과를 평가할 수 없음
연결할 사내 시스템에 API가 있는가	MCP·스크립트로 기록까지 자동 처리	결과 문서 작성까지만 맡김
사람이 승인할 단계가 하나 이상 있는가	잘못된 결과가 외부에 나가기 전에 멈춤	승인 단계를 먼저 설계

12.3.1 견적 작성과 닮은 후보 업무의 특징

견적 작성 업무가 첫 업무에 맞는 이유

6장의 견적 작성은 세 조건을 모두 갖춘 업무입니다. 공공기관 견적은 VAT 포함, 조달 품목은 할인 없이 등록가 그대로, 조달 등록가는 VAT 포함가이므로 품목 단가는 등록가 ÷ 1.1처럼 규칙이 문장으로 정리되고, 견적 요청

은 매주 반복되며, 고객 이름과 금액은 조직 밖으로 나가면 안 되는 자료입니다. 그래서 OPENMARU AAS 채팅에서 견적 PDF 요청은 10초 안팎에 처리되고, 이 과정에서 모델이 도구를 직접 부르지 않고 미리 만든 코드가 판단합니다. 또한 작성 전과 작성 후 두 번 점검해, 공공기관인데 VAT 별도로 작성된 견적이나 조달 견적에 할인이 걸린 견적은 만들기 전에 멈춥니다.

견적 작성과 닮은 업무 범주

같은 특징을 가진 업무는 부서마다 있습니다. 영업·구매에서는 견적서와 제안 가격표 작성, 발주서와 계약 조건 대조, 단가표 갱신이 후보이고, 재무·회계에서는 증빙 서류의 필수 항목 점검과 정해진 양식의 정기 정산 보고가 후보입니다. 인사·총무의 지원 서류 요건 점검과 신청서 1차 검토, 공공 민원·문의의 유형 분류와 담당 부서 지정, 규정 조항을 찾아 쓰는 답변 초안도 같은 범주에 들어갑니다. IT 운영에서는 예약 작업 실패 확인과 담당자 알림, 점검 결과의 정기 보고 작성이 12.3의 점검표에서 「예」가 많이 나오는 업무입니다.

이 범주의 공통점은 규칙이 이미 규정집이나 업무 지침에 적혀 있고, 결과물이 정해진 양식의 문서라는 점입니다. 반대로 매번 판단 근거가 바뀌는 기획 업무나 결과가 맞는지 판정할 수 없는 창작 업무는 첫 업무로 고르지 않습니다.

결과물이 조직 시스템에 쌓이는 업무부터

후보가 여럿이면 일을 시작하게 하는 신호와 사람이 판단하는 단계가 분명하고, 결과물이 위키·CRM·문서 저장소처럼 조직이 이미 쓰는 시스템에 쌓이는 업무부터 고르는 것이 판단하기 쉽습니다. 시작 신호가 분명하면 예약 작업이나 채팅 요청으로 일을 맡기기 쉽고, 사람이 판단하는 단계가 분명하면 결과를 누가 책임지는지가 정해집니다. 결과물이 조직 시스템에 쌓이면 시범 기간이 끝난 뒤에도 처리 건수와 수정 이력을 그대로 효과 측정의 근거로 쓸 수 있습니다. 명함 정리, 회의록 작성, 정해진 서식 채우기처럼 세 조건을 함께 갖춘 업무는 견적 작성과 함께 첫 업무 후보로 올려 볼 만합니다.

12.3.2 첫 업무 하나의 시범 도입 성공 판정 기준

성공·확대·중단을 가르는 판정표

시범 도입의 성공 판정 기준은 시작 전에 정해야 하며, 10.4의 비교 지표를 업무 하나에 맞춰 쓰면 됩니다. 판정은 성공, 확대, 중단의 세 갈래로 나누고, 각 갈래마다 의사결정권자가 다음 결재를 무엇으로 할지 함께 정합니다.

판정	기준	다음 결재
성공	요청 한 건의 처리 시간이 시범 전보다 줄고, 점검 단계가 규칙 위반을 만들기 전에 멈추며, 업무 담당자가 계속 쓰기로 함	같은 부서의 두 번째 업무 선정
확대	성공 조건에 더해 다른 부서가 같은 스킬을 쓰겠다고 요청하고, 규칙을 바꿀 때 업무 담당자가 말로 고친 내용이 코딩 에이전트를 거쳐 반영됨	사내 허브 게시와 4단계 부서 확산
중단	결과를 매번 사람이 다시 만들거나, 자료가 조직 밖으로 나가는 경로가 확	1단계로 돌아가 후보 업무 재선정

판정	기준	다음 결재
	인되거나, 규칙을 문장으로 정하지 못함	

시범 기간은 날짜보다 처리 건수로 정하는 편이 판정에 유리합니다. 견적처럼 매주 반복되는 업무라면 업무 담당자가 규칙을 한두 번 고쳐 다시 반영해 볼 수 있을 만큼의 요청이 쌓여야 성공과 중단을 가를 근거가 생기기 때문입니다. 그래서 결재 문서에는 「요청 몇 건을 처리한 뒤 판정한다」는 조건과 그 사이 규칙을 몇 번까지 고칠 수 있는지를 함께 적어 둡니다.

판정에서 함께 확인할 제품 동작

성공 판정에는 숫자와 함께 제품이 실제로 어떤 방식으로 일했는지도 기록합니다. 판단을 미리 만든 코드가 맡았는지, 새 규칙을 지난 처리 결과 전체에 다시 대어 통과·경고·차단 건수를 볼 수 있었는지, 모든 모델 호출이 게이트웨이를 거쳤는지가 그 항목입니다. 이 세 가지가 확인되면 처리 속도가 같더라도 업무 규칙이 사람에게서 조직의 스킬로 옮겨 갔다고 판단할 수 있고, 그 스킬은 담당자가 바뀌어도 조직에 남습니다.

12.4 AI 단계·내부 sLLM 필요성·제작 주체·첫 업무 — 의사결정권자를 위한 도입 결재 점검표

OPENMARU AAS 도입 결정은 네 가지 판단으로 모입니다. 우리 조직의 AI가 지금 어느 단계인지, 그 다음 단계를 내부 sLLM으로 가야 하는지, AI Agent를 누가 만들게 할지, 시범 도입할 첫 업무가 무엇인지입니다. 앞 장들이 각 판단의 근거를 표로 정리했으므로, 이 절은 그 결과를 결재 문서 한 장으로 옮기고 IT 담당자가 준비할 점검 항목을 함께 둡니다. 네 판단에 모두 답이 있으면 의사결정권자는 이 한 장으로 시범 도입을 결재할 수 있습니다.

12.4.1 의사결정권자가 내릴 네 가지 판단과 근거 장 번호

도입 결재 점검표

아래 표는 결재 문서에 그대로 붙여 쓸 수 있도록 만든 점검표이며, 각 판단 칸의 근거는 오른쪽 장 번호의 판단표에 있습니다.

판단	결재 질문	선택지	근거 장	결정
1. AI 단계	우리 조직의 AI는 Prompt·Context·Loop·AI Agent·Agentic AI 가운데 어디에 있고, 다음 단계는 무엇인가	현재 단계 / 다음 단계	3장	☐
2. 내부 sLLM 필요성	다음 단계에서 다룰 자료를 외부 LLM에 보낼 수 있는가, 폐쇄망 안에서 돌려야 하는가	외부 LLM 허용 / 내부 sLLM 필요 / 업무별로 나눔	4장	☐

판단	결재 질문	선택지	근거 장	결정
3. 제작 주체	AI Agent를 AI 개발자에게 맡길 것인가, 업무 담당자가 말로 정의하고 코딩 에이전트가 만들게 할 것인가	개발 부서 / 업무 담당자와 IT 담당자 짝	5장	☐
4. 첫 업무	업무 담당자가 말로 정의할 첫 업무 하나는 무엇이고, 성공 판정 기준은 무엇인가	업무 이름 / 성공·확대·중단 기준	6장, 12.3	☐

판단을 뒷받침하는 보조 확인

네 판단 외에 두 가지를 보조로 확인합니다. 하나는 11장의 라이선스와 공급사 종속 위험이며, 코드는 Apache-2.0 이고 이름·로고와 제3자 구성요소는 별도로 검토했다는 문장이면 됩니다. 다른 하나는 12.1의 남은 한계 가운데 시범 도입을 막는 항목이 있는지입니다. 두 확인이 끝나고 네 판단에 모두 답이 있으면 결재 범위는 「한 부서, 한 업무, 정해진 기간의 시범 도입」으로 좁혀지며, 이 범위가 도입 4단계의 3단계에 해당합니다.

12.4.2 IT 담당자가 준비할 모델 게이트웨이·권한·연동 점검 항목

모델 게이트웨이와 설치 점검 항목

IT 담당자의 준비 항목은 4.3의 폐쇄망 점검표와 9.3의 권한 표를 시범 도입 범위로 묶은 것입니다. 모델 쪽에서는 사내 모델이 도구 호출과 64,000 토큰 이상의 문맥을 지원하는지, OpenAI 호환 주소 하나로 연결되는지를 먼저 확인합니다[S14]. 관리자 매뉴얼에 따르면 대화용 모델 목록에는 운영자가 정한 이름 규칙에 맞는 게이트웨이 모델만 보이고, 질문 가드레일과 위험한 명령 차단의 두 안전장치는 화면에서 끌 수 없습니다[S13].

영역	점검 항목	근거
모델 게이트웨이	모든 모델 호출이 LiteLLM 같은 게이트웨이를 거치고 예산 한도가 걸려 있는가	4.3
설치 형태	시범 도입은 Docker Compose 공개본, 부서 확산은 쿠버네티스 구성으로 정했는가	11.3
설치 형태	폐쇄망 내부 레지스트리와 데이터 볼륨 백업 위치를 정했는가	11.3
권한	사내 로그인(SSO)을 연결하고 업무 담당자·IT 담당자에게 맞는 권한을 줬는가	9.3
권한	비밀값을 입력하는 화면을 HTTPS로 열었는가	공개 저장소

영역	점검 항목	근거
연동	첫 업무가 쓰는 사내 시스템의 API를 MCP 나 스크립트로 부를 수 있는가	6장
연동	만드는 단계의 코딩 에이전트를 외부 서비스로 쓸지, 사내 코딩 모델로 쓸지 정했는가	6장

점검표를 끝낸 뒤의 첫 결재

이 점검표가 채워지면 IT 담당자는 12.4.1의 네 판단에 기술 근거를 붙일 수 있고, 의사결정권자는 우리 조직의 AI 단계와 내부 sLLM 필요성을 한 번에 판단할 수 있습니다. 남는 결정은 업무 담당자가 말로 정의할 첫 업무 하나를 골라 OPENMARU AAS 시범 도입 대상으로 정하는 일이며, 그 업무에서 정한 규칙과 스킬이 다음 부서로 넓혀 갈 조직의 첫 자산이 됩니다. 말하는 챗봇에서 일하는 AI Agent로 가는 길은 더 큰 모델이나 새 개발 조직보다, 업무 담당자가 말로 정의한 업무 하나를 조직 안의 sLLM 위에서 일하게 하는 데서 시작합니다.

부록 A. 용어 풀이

본문에 나오는 주요 용어를 처음 나오는 순서대로 모았습니다.

용어	정의
AI Agent(AI 에이전트)	언어 모델이 도구·기억·권한을 갖추고 실행 경로를 스스로 정하면서 업무 한 건을 결과물까지 처리하는 시스템
RAG(Retrieval-Augmented Generation)	질문과 관련된 사내 문서를 먼저 검색해 답의 근거로 붙이는 방식. 사내 챗봇이 주로 쓴다
워크플로	사람이 미리 작성한 코드 경로를 따라 언어 모델과 도구가 정해진 순서로 실행되는 시스템
LLM(Large Language Model, 대규모 언어 모델)	많은 양의 글로 학습해 자연어를 이해하고 만들어 내는 언어 모델
Agentic AI(에이전틱 AI)	여러 AI Agent가 역할을 나눠 서로 조율하며 일하는 구성
Prompt 단계	LLM에게 원하는 것을 자연어로 잘 적어 답을 받는 단계
Context(컨텍스트) 단계	질문과 관련된 사내 자료를 먼저 찾아 모델 입력에 함께 넣어 주는 단계. 답은 정확해지지만 결과를 옮겨 적는 일은 사람에게 남는다
Loop	AI가 계획·실행·결과 확인·재시도를 업무가 끝날 때까지 스스로 반복하는 단계
ReAct(Reasoning and Acting)	언어 모델이 추론과 행동을 번갈아 생성하고, 행동 결과를 다음 추론에 반영하도록 구성한 방식
MCP(Model Context Protocol)	언어 모델이 외부 도구와 데이터에 접속하는 표준 규약. 업무 시스템 기능을 에이전트의 도구로 등록할 때 쓴다
PoC(Proof of Concept, 개념 검증)	본 도입 전에 기술이 업무에 통하는지 작은 범위에서 확인하는 시험 사업
GPU	언어 모델을 학습시키거나 서빙할 때 쓰는 연산 장비
sLLM(small Large Language Model)	조직 안 장비에서 직접 서빙할 수 있는 수십억–수백억 파라미터 규모의 언어 모델
폐쇄망	외부 인터넷과 분리해 운영하는 조직 내부 네트워크. 실행 중 외부로 나가는 요청이 없어야 한다
CRM(Customer Relationship Management)	고객·영업기회·견적 같은 영업 정보를 관리하는 업무 시스템
API	프로그램끼리 기능과 자료를 주고받도록 정해 둔 호출 방식

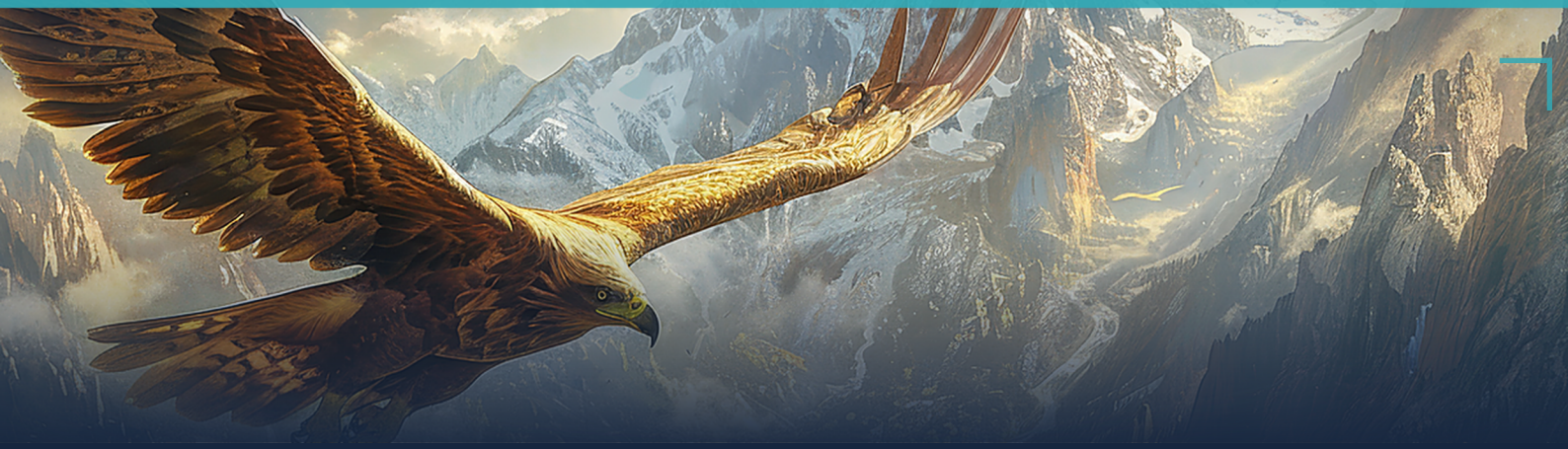
용어	정의
승인 받은편지함	에이전트가 사람의 승인이나 답을 기다리는 요청을 한곳에 모아 보여 주는 화면
스킬	자주 하는 일의 순서와 주의점을 적어 둔 업무 방법. AI 에이전트는 같은 일을 맡을 때마다 이 순서를 따르며, OPENMARU AAS에서는 규칙 파일이 정본이다
작업공간	에이전트 실행 환경 하나에 해당하는 사용자별 단위. 프로파일을 하나 이상 가지며 비밀값과 권한을 따로 관리한다
코딩 에이전트	업무 담당자가 말한 규칙 문장을 규칙 파일·점검 코드·시험으로 옮기는 코드 작성용 AI 에이전트
모델 게이트웨이	여러 모델 서버 앞에서 모델 호출을 받아 넘겨주는 중계 서버. 모든 호출과 기록을 한곳에서 관리한다(예: LiteLLM)
OpenAI 호환 주소	OpenAI API와 같은 형식으로 요청을 받는 모델 서버 주소. vLLM·LiteLLM·Ollama가 이 형식을 지원한다
토큰	언어 모델이 글을 나눠 처리하는 단위. 문맥 길이와 사용량을 셀 때 쓴다
Hermes Agent	OPENMARU AAS 아래에서 에이전트를 실제로 실행하는 오픈소스 에이전트 플랫폼
예약 작업	정해진 시각에 AI 에이전트가 같은 일을 자동으로 하도록 등록한 작업
스킬 허브	사내에서 쓰는 스킬 패키지를 모아 두고, 관리자 게시 검토를 거쳐 다른 부서가 설치해 쓰게 하는 곳
Apache-2.0	코드의 사용·수정·재배포를 허락하는 오픈소스 라이선스. 저작권 고지를 유지해야 하며 회사 이름과 로고 사용은 허락 범위에 들어가지 않는다
프로파일	AI 에이전트의 성격, 쓸 모델, 접속할 외부 서비스 같은 설정과 권한을 모아 둔 단위
SSO(Single Sign-On)	사내 로그인 하나로 여러 시스템에 들어가는 통합 로그인 방식
SaaS(Software as a Service)	조직 안에 설치하지 않고 공급사의 클라우드에서 서비스로 쓰는 소프트웨어

부록 B. 참고문헌

본문의 [S] 번호와 같은 순서로 실었습니다.

1. [S1] McKinsey & Company. (2025-11-05). "The state of AI in 2025: Agents, innovation, and transformation". https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai
2. [S2] McKinsey & Company. (2025-11). "AI at work but not at scale (Week in Charts)". https://www.mckinsey.com/featured-insights/week-in-charts/ai-at-work-but-not-at-scale
3. [S3] Gartner. (2024-07-29). "Gartner Predicts 30% of Generative AI Projects Will Be Abandoned After Proof of Concept By End of 2025". https://www.gartner.com/en/newsroom/press-releases/2024-07-29-gartner-predicts-30-percent-of-generative-ai-projects-will-be-abandoned-after-proof-of-concept-by-end-of-2025
4. [S4] Gartner. (2025-06-25). "Gartner Predicts Over 40% of Agentic AI Projects Will Be Canceled by End of 2027". https://www.gartner.com/en/newsroom/press-releases/2025-06-25-gartner-predicts-over-40-percent-of-agentic-ai-projects-will-be-canceled-by-end-of-2027
5. [S5] Gartner. (2024-10-21). "AI Agents (Top Strategic Technology Trends for 2025: Agentic AI)". https://www.gartner.com/en/articles/ai-agents-pc1
6. [S6] Gartner. (2024-10-21). "Gartner Identifies the Top 10 Strategic Technology Trends for 2025". https://www.gartner.com/en/newsroom/press-releases/2024-10-21-gartner-identifies-the-top-10-strategic-technology-trends-for-2025
7. [S7] Gartner. (2025-08-26). "Gartner Predicts 40% of Enterprise Apps Will Feature Task-Specific AI Agents by 2026, Up from Less Than 5% in 2025". https://www.gartner.com/en/newsroom/press-releases/2025-08-26-gartner-predicts-40-percent-of-enterprise-apps-will-feature-task-specific-ai-agents-by-2026-up-from-less-than-5-percent-in-2025
8. [S8] Omdia. (2025-09-22). "New Omdia Analysis Shows Agentic AI Outpacing Growth Rates of Traditional Generative AI". https://omdia.tech.informa.com/pr/2025/sep/new-omdia-analysis-shows-agentic-ai-outpacing-growth-rates-of-traditional-generative-ai
9. [S9] Gartner. (2025-07-08). "The Latest Hype Cycle for Artificial Intelligence Goes Beyond GenAI". https://www.gartner.com/en/articles/hype-cycle-for-artificial-intelligence
10. [S10] Yao, S. 외. (2022-10-06, ICLR 2023 발표). "ReAct: Synergizing Reasoning and Acting in Language Models". https://arxiv.org/abs/2210.03629

11. [S11] Anthropic. (2024-12-19). "Building effective agents". https://www.anthropic.com/research/building-effective-agents
12. [S12] OPENMARU. (2026-11 접속). "AAS 사용자 매뉴얼". https://docs.openmaru.io/docs/aas-user-guide/aas-intro/
13. [S13] OPENMARU. (2026-11 접속). "AAS 관리자 매뉴얼". https://docs.openmaru.io/docs/aas-admin-guide/admin-overview/
14. [S14] OPENMARU. (2026-11-07). "ai-agent-studio 공개 저장소 (README·LICENSE·NOTICE·TRADEMARKS·THIRD_PARTY_NOTICES)". https://github.com/openmaru-io/ai-agent-studio



말하는 챗봇에서 일하는 AI Agent로

CONTACT

WEB
msap.ai
www.msap.ai/

EMAIL
hello@msap.ai

TEL
02-6953-5427
0269535427

YOUTUBE
@msaptv
www.youtube.com/@msaptv

LINKEDIN
linkedin.com/showcas...
www.linkedin.com/showcase/msap-ai/

FACEBOOK
facebook.com/opennaru
www.facebook.com/opennaru

SCAN

© 2026 SMART CF, MSAP.ai. All rights reserved.