LAURENT DÉSÉCHALLIERS

컨텍스트

AI가 중요한 것을 이해하고,
지식을 연결하며,
배움을 목표에 활용할 수 있는
컨텍스트를 만드세요.

Le Contexte

> *AI*가 중요한 것을 이해하고, 지식을 서로 연결하며, 목표를 위해 배운 것을 자산으로 쌓아 갈 수 있는 컨텍스트를 어떻게 만들 것인가.

Laurent Deséchalliers

번역에 관한 안내

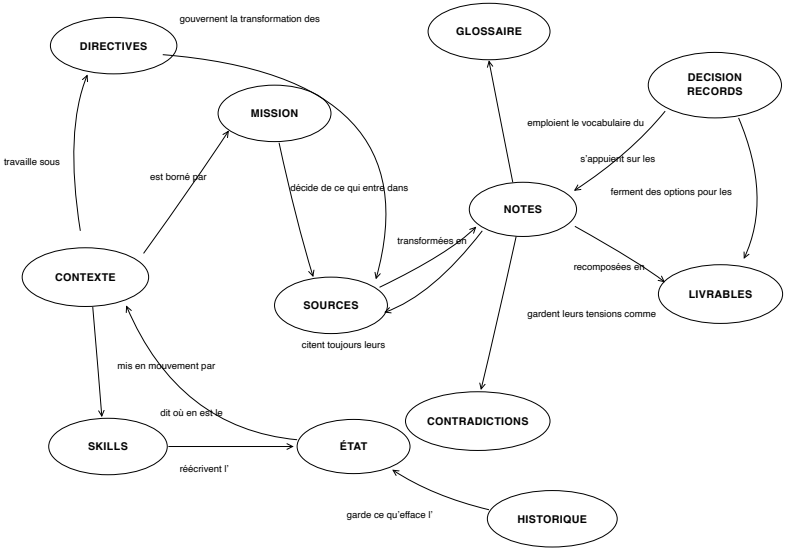
이 판본은 프랑스어 원문을 바탕으로 인공지능이 자동 번역했습니다.

인간 번역가의 검수를 거치지 않았습니다. 아이디어와 구조, 예시는 원문에 충실하지만, 일부 표현이 어색하거나 몇몇 대목에 원문인 프랑스어의 흔적이 남아 있습니다.

이는 이 책을 프랑스어로만 남겨 두지 않고 여러 언어로 제공하기 위해 감수한 대가입니다. 표현이 불분명할 때는 프랑스어판을 기준으로 삼아 주십시오.

컨텍스트의 정의

> 컨텍스트는 사람의 임무로 범위가 정해진 시스템입니다. 이 안에서 *AI*는 우리가 알고 있는 것, 그 지식의 출처, 각 요소의 관계, 그리고 현재의 작업 상태를 정리하고 활용합니다. 그로써 사람이 매번 처음부터 다시 시작하지 않고도 목표에 다가갈 수 있게 돕습니다. 이 과정에서 시스템은 오래 남는 정보 자산을 만들고 키웁니다. 생성된 지식은 계속 사용할 수 있고, 시간이 흐르며 풍부해지고, 다른 목표에도 다시 활용됩니다.



서문

우리는 지금껏 이렇게 많은 정보를 이용할 수 있었던 적이 없습니다. 그럼에도 며칠 전에 이해했던 것을 찾고, 다시 정리하고, 처음부터 재구성하는 데 여전히 많은 에너지를 씁니다.

쓸모 있는 아이디어는 깔끔한 모습으로 나타나는 일이 드뭅니다. 대화에서 태어나 문서 안에서 구체화되고, 회의에서 반론을 만나며, 결정으로 이어진 뒤 글이나 명세서, 프로젝트가 됩니다. 우리의 도구는 이 흔적들을 하나하나 보관합니다. 하지만 그 사이를 잇는 흐름은 잘 지키지 못합니다.

인공지능은 이 문제의 일부를 바꿔 놓았습니다. AI는 서로 다른 종류의 자료를 읽고, 연결하고, 다시 표현하고, 변환할 수 있습니다. 그러나 AI와의 대화는 여전히 흐름이 쉽게 끊깁니다. 주제는 앞으로 나아가고 대화는 길어지지만, 결국 목적과 이미 내린 결정, 사용한 출처, 남은 일을 다시 설명해야 합니다.

이 어려움을 해결하기 위해 저는 컨텍스트를 만들었습니다.

컨텍스트는 하나의 주제에 전념하는 작업 공간입니다. 사람과 AI는 이 안에 추구하는 목적, 출처, 아이디어, 규칙, 작업 상태를 보관합니다. 그래서 매 세션마다 모든 것을 다시 만들지 않고도 결과물을 낼 수 있습니다.

그러면 AI는 실제 기록을 바탕으로 일할 수 있습니다. 아이디어의 출발점을 찾고, 선택을 상기시키고, 모순을 알리고, 부족한 근거를 짚어 내며, 쓰다 멈춘 초안을 그 지점에서 다시 이어 갑니다. AI는 생각을 정리하고 검증하는 일을 돕습니다. 의미와 결정에 대한 책임은 사람에게 남습니다.

이 시스템은 제 매일의 업무 속에서, 파일 하나와 AI에게 던진 아주 간단한 요청에서 태어났습니다. 제가 앞으로 하는 말을 정리해 달라는 요청이었습니다. 이후 시행착오를 거치며, 문제가 하나 드러날 때마다 빈자리를 채워 가듯 발전했습니다. 지금도 계속 바뀌고 있습니다. 여기서 소개하는

해법 중에는 여러 용도에서 견딘 것도 있고, 아직 마주하고 있는 문제에 대한 임시 답안도 있습니다.

따라서 이 책은 완성된 이론에서 출발하지 않습니다. 현실에서 겪은 문제들, 즉 더 이상 따라갈 수 없게 된 500개의 이메일, 여러 도구에 흩어진 아이디어, 이미 한 일을 잃지 않은 채 답해야 했던 고객 문서, 중단 후 다시 쓰기 시작한 글에서 출발합니다.

이 책은 정보를 이해와 결정, 결과물로 바꾸는 사람들을 위한 책입니다. 프로덕트 매니저, 컨설턴트, 글쓰는 사람, 연구자, 분석가, 그리고 더 넓게는 지식 노동자를 대상으로 합니다. 자신의 정보를 통제하면서 AI를 오랫동안 함께 일하는 동료로 만드는 방법을 제안합니다.

이 책은 서로 보완하는 두 권으로 구성됩니다. **1**권 — **컨텍스트** 이해하기는 시스템이 태어난 과정과 AI와의 관계를 어떻게 바꾸는지를 이야기합니다. **2**권 — **컨텍스트** 만들고 유지하기는 이를 어떻게 만들고 오랫동안 사용하는지 설명합니다.

두 권을 순서대로 읽어도 되고, 따로 읽어도 됩니다. 이 책을 어떤 순서로 읽을까 장에서 여러 가지 읽기 경로를 소개합니다.

예시에 관한 안내

이 책의 생각과 예시는 30년이 넘는 IT 업계 경험에서 나왔습니다. 서로 다른 프로젝트와 업무 환경에서 겪은 상황, 그리고 다른 회사의 많은 프로덕트 매니저, CTO, 엔지니어링 매니저 등과 나눈 대화가 바탕이 되었습니다. 여러 조직과 제 커리어의 여러 시기에서 쌓인 경험입니다.

책의 논지를 위해 이 상황들을 수정하고, 일반화하거나, 여러 사례를 재구성했습니다. 사용 사례, 수치, 지표는 반복해서 나타나는 문제와 그로부터 얻은 교훈을 보여 주기 위한 구성입니다. 실제 고객, 제품, 조직을 묘사하지 않으며, 제 현직장의 데이터, 업무 방식, 입장을 반영하지도 않습니다.

공동 주택, 제 글과 저작물 등 개인적인 삶에서 가져온 예시는 제가 직접 겪은 상황을 바탕으로 합니다. 다만 특정 인물을 알아볼 수 있는 세부 사항은 일부 바꾸었습니다.

이 책에 담긴 분석, 방법, 견해는 제 개인의 것이며 오직 제가 책임집니다.

저자 소개

저는 수십 년 동안 소프트웨어 산업에서 일해 왔고, 현재는 프로덕트 매니저로 일합니다. 개발과 아키텍처에서 시작해 팀 운영으로, 다시 제품과 그 운영, 마케팅으로 영역을 넓혀 왔습니다.

업무가 바뀌면서 정보를 대하는 방식도 바뀌었습니다. 기술적인 해법을 알거나 문서 하나를 찾는 것만으로는 부족했습니다. 흩어져 있는 신호를 모아 함께 무엇을 말하는지 이해하고, 여러 방향 사이에서 선택하며, 서로 다른 독자에게 맞는 글을 만들어야 했습니다.

인공지능과 분산 시스템에 대한 제 관심은 오래되었습니다. 오늘날 그 관심은 아주 구체적인 질문을 만납니다. 자신의 정보와 결정, 생각을 통제할 권한을 **AI**에게 넘기지 않으면서 어떻게 **AI**와 함께 일할 것인가**?**

저는 전문 언론에 기여했고 M2 Édition과 Eyrolles에서 기술 감수자로 일했습니다. 지금은 malorean.net에서 프로덕트 매니지먼트, AI, 일하는 방법, 정보 시스템에 관해 글을 쓰고 있습니다.

전략 및 실무 마케팅을 다루는 독립 잡지 *Proposition Marketing*에도 기고합니다. 마케팅과 기업 경영 분야의 독자를 위해 쓰다 보면 공통의 용어를 안다고 가정하지 않고 제품에 관한 문제를 설명해야 합니다. 이 책도 같은 원칙을 따릅니다.

비슷하거나, 다르거나, 서로 보완하는 시스템을 만드는 다른 분들과 기쁘게 이야기를 나누고 싶습니다. 이 주제는 논의하고, 검증하고, 다른 실천으로 풍부하게 만들 가치가 있습니다. 인접한 주제를 연구하는 학계의 분들과도 교류하고 싶습니다. 저는 프랑스 국립과학연구센터(CNRS) 산하 두 연구소에서 몇 년을 보냈고, 그 세계와 다시 인연을 맺는다면 기쁘겠습니다.

Laurent Deséchalliers

- laurent@malorean.net

- https://malorean.net

감사의 말

가장 먼저 감사드릴 분은 독자 여러분입니다.

예상보다 조금 두껍고, 반복되는 내용도 아마 조금 있으며, 다소 난해한 주제를 다루는 이 책을 펼쳐 보겠다는 조금은 무모한 생각을 해 주신 여러분께 감사드립니다.

지금껏 저는 비교적 혼자 컨텍스트를 만들어 왔습니다. 그러나 이 고독함이 오래가지 않기를 바랍니다. 다른 사람들도 이 주제를 탐구하고, 논의하고, 비판하고, 발전시키고 싶어지면 좋겠습니다. 그렇게 된다면 앞으로 malorean.net에 올릴 글에서 많은 기여자들에게 감사를 전할 수 있을지도 모릅니다.

그리고 무엇보다 제 아내 Christine에게 감사합니다.

아내는 제가 이 책을 붙들고 밤늦게까지, 주말 내내 보내는 모습을 지켜봤습니다. 쓰는 내내 저를 지지해 주었고, 너무 늦었으니 이제 정말 그 지겨운 컴퓨터를 덮을 때가 됐다고 제대로 일깨워 주기도 했습니다.

약속할게요. 이건 잠깐의 예외였어요.

그리고 이 책을 훨씬 넘어서, 이렇게 멋진 삶을 오랫동안 함께해 줘서 고마워요.

이 책을 어떤 순서로 읽을까

이 책은 일이 저에게 실제로 일어난 순서를 따릅니다. 먼저 문제를 겪고, 임시방편으로 해법을 만들었으며, 마침내 하나의 시스템에 이르렀습니다. 그래서 컨텍스트의 전체 설명은 2권에서야 나옵니다.

'이해하기' 경로는 바로 이 순서입니다. 처음부터 끝까지 읽으면, 각 구성 요소는 그것을 필요하게 만든 빈자리가 드러난 직후에 등장합니다.

'만들기' 경로는 이미 같은 문제를 겪었고, 이야기보다는 작동 방식을 먼저 알고 싶은 사람을 위한 경로입니다.

1. **컨텍스트**의 정의: 무엇에 관한 이야기인지 알기 위한 몇 줄입니다.
2. **5**부 — **컨텍스트** 안을 들여다보기: 여섯 장의 순서대로 지도, 열기, 자료 넣기, 변환하기, 생산하고 결정하기, 저장하기를 읽습니다.
3. **6**부 — 생태계 안의 **컨텍스트**: 방금 만든 것을 정보 시스템의 다른 요소와 연결합니다.
4. **1**권: 설계 선택이 임의적으로 보일 때 읽으면, 그 선택을 만든 문제를 찾을 수 있습니다.

이 지름길에는 대가가 있습니다. 2권은 1권의 상황을 이미 알고 있는 독자를 상정해 쓰였고, 그 상황을 다시 언급합니다. 따라서 맥락이 조금 부족한 예시를 만나게 됩니다. 그것이 이 경로의 대가지만, 왜 이 책을 펼쳤는지 이미 알고 있다면 여전히 감수할 만합니다.

읽기 전에 시험해 보기

이 책은 하나의 방법을 설명하지만, 방법은 말로만 설명하기 어렵습니다. 이론을 읽기 전이나 읽는 중에 실제로 작동하는 모습을 볼 수 있습니다. 각 구성 요소의 쓸모를 이해하기에는 그게 가장 좋은 순간일 때가 많습니다.

github.com/laurent-malorean에 세 개의 저장소가 공개되어 있습니다.

Contexte Nano Notion은 가장 빠르게 설치할 수 있습니다. 개발 환경도, 기술 지식도 필요 없고 ChatGPT와 Notion만 있으면 됩니다. ChatGPT는 컨텍스트의 지능을 담당하고, Notion은 기억을 담당합니다. 출처, 노트, 결정, 상태가 여기에 기록되고 대화가 바뀌어도 다시 읽힙니다. 몇 분이면 작동하는 컨텍스트를 갖출 수 있습니다.

Contexte V2는 완전한 구현입니다. 제가 매일 사용하고 이 책을 쓰는 데 사용한 버전입니다. Claude Code와 Codex 중 하나를 골라 설치할 수 있으며, 앞의 버전보다 준비가 조금 더 필요합니다.

LLM 위키는 컨텍스트가 아닙니다. 책, 표준, 글, 강의 등의 문서로 만든 지식 라이브러리입니다. AI가 문서에서 짧은 지식 카드를 만들고 서로 연결하며, 이후 자연어로 질문할 수 있습니다. 컨텍스트는 하나의 임무가 이어지는 흐름을 지키고, LLM 위키는 우리가 읽은 것을 대규모로 보관합니다. 6부에서 둘이 어떻게 함께 일하는지 설명합니다.

반드시 여기서 시작할 필요는 없습니다. 하지만 읽다가 내용이 추상적으로 느껴진다면, 다시 구체적으로 이해하는 가장 짧은 길입니다.

같은 저자의 책

Malorean Corp — 어제까지만 해도 하나의 아이디어를 프로토타입으로 만드는 데 몇 주가 걸리곤 했습니다. 오늘날에는 인공지능의 도움을 받는 한 사람이 며칠 만에 해낼 수 있습니다. 놀라운 변화입니다. 하지만 진정한 격변은 다른 곳, 바로 우리가 함께 일하는 방식에서 일어날지도 모릅니다.
출간 예정: *2026*년 *11*월 말

자세한 내용: https://malorean.net.

목차

권 I

컨텍스트 이해하기

부 1 – 필요의 탄생 19
1.1 500개의 이메일과 사라지는 경위 20
1.2 첫 번째 정의 27

부 2 – 컨텍스트가 바꾸는 것 33
2.1 대화에서 다시 이어갈 수 있는 작업으로 34
2.2 두 고객, 하나의 백서, 그리고 보존해야 할 모순 40

부 3 – 세 가지 구체적인 활용법 49
3.1 이미 해 놓은 일을 잃지 않고 제안요청서에 답하기 50
3.2 아직 형태를 갖추지 못한 생각으로 글쓰기 55
3.3 혼란 속에서 제품 주제를 발전시키기 61

부 4 – 결과물이 남기고 간 것 69
4.1 답은 셀 하나에 담기지만 추론은 담기지 않는다 70
4.2 6개월 뒤, 답만으로는 충분하지 않다 73

권 II

컨텍스트를 만들고 살아 있게 하기

부 5 – 컨텍스트 안을 들여다보기 79
5.1 전체 지도 80
5.2 열기: 임무와 현재 상태 되찾기 90
5.3 자료 넣기: 재료를 받아들이기 98
5.4 변환하기: 출처에서 아이디어로 103
5.5 결과물을 만들고 결정하기 110
5.6 저장하고 재개하기 117

부 6 – 생태계 안의 컨텍스트 123
6.1 구조화된 지식에 질문하기 124
6.2 실제 제품에 질문하기 129
6.3 컨텍스트가 세 개가 된 날 136
6.4 이웃집 자료를 읽되 그 집으로 이사 가지 않기 141
6.5 지식 순환의 대가 147

책을 덮기 전에

맺음말 153
7.1 에필로그 154
7.2 계보와 이웃한 접근법 156

권 I

컨텍스트 이해하기

1 필요의 탄생
2 컨텍스트가 바꾸는 것
3 세 가지 구체적인 활용법
4 결과물이 남기고 간 것

부 1

필요의 탄생

컨텍스트는 인공지능에 대한 아이디어에서 태어나지 않았습니다. 더 이상 쓸 수 없게 된 메일함과, 서서히 모습을 드러낸 하나의 깨달음에서 태어났습니다. 문제는 메시지의 수가 아니었습니다.

이 부에서는 그 시작을 이야기한 뒤 첫 번째 정의를 끌어냅니다. 기억하기에는 충분히 간단하고, 책의 나머지 부분을 이끌 기준이 되기에는 충분히 완전한 정의입니다.

1.1

500개의 이메일과 사라지는 경위

제가 처음 컨텍스트를 만들었을 때, 인공지능과 함께 일하는 방법을 발명하려던 것은 아닙니다. 공동주택에서 일어나는 일의 흐름을 더 이상 놓치지 않으려고 했을 뿐입니다.

저는 공동소유자 협의회에 참여합니다. 그 자격으로 관리업체, 공동소유자, 임차인, 용역업체가 보내는 메시지를 받습니다.

- 한 이메일은 새로운 문제를 알립니다.
- 다음 이메일은 견적의 세부 내용을 보충합니다.
- 세 번째 이메일은 작업 일정을 미룹니다.
- 일주일 뒤, 누군가가 같은 대화 스레드에서 다른 건물 이야기를 꺼냅니다.

그 사이에 다른 세 가지 안건도 앞으로 나아갑니다.

문제는 양만이 아니었습니다. 약 500개의 이메일이 15개 안팎의 주제에 흩어져 있었습니다.

- 공사,
- 재정,
- 보험,
- 총회,
- 미납자,
- 입찰,
- 사고와 기타 요청
- …

각 주제는 자기만의 속도로, 때로는 몇 달에 걸쳐 진행됐습니다.

매일 공사를 따라가는 사람은 자연스럽게 그 경위를 기억합니다. 어느 업체가 작업했는지, 일정이 왜 바뀌었는지, 어느 결정이 아직 답을 기다리는

지 알고 있습니다. 새 메시지를 열었을 때 저에게는 항상 그런 연속성이 있지 않았습니다. 새로운 정보를 이해하기도 전에 스레드를 거슬러 올라가 이전 대화를 찾고 상황을 다시 조립해야 했습니다.

저는 간단한 질문을 할 수 있기를 원했습니다.

- 지붕 공사는 지금 어디까지 진행됐나?
- 물받이 관련 최신 견적을 보낸 업체는 어디인가?
- 이 누수에 대해 우리는 무엇을 결정했나?
- 관리업체가 아직 해야 할 일은 무엇인가?

이메일함 안에 답은 있었습니다. 하지만 전체 그림은 보여 주지 않았습니다.

파일 하나와 한 문장

저는 가능한 한 가장 간단한 구성으로 시작했습니다. 파일 하나와 AI에게 준 지시 하나였습니다.

> 정보를 모아 주제별로 나눈다.

저는 AI에게 이메일을 넘겼습니다. AI는 이메일을 주제와 연결하고 주제별 요약을 계속 갱신했습니다.

- 공사에 관한 정보는 한곳에 모였습니다.
- 재정 문제는 또 다른 묶음을 이뤘습니다.
- 업체, 보험, 총회도 각자 자리를 찾아갔습니다.

이 첫 시도만으로도 이미 얻는 것이 있었습니다. 관련 메시지를 모두 다시 읽지 않고도 주제의 현재 상태를 물을 수 있었습니다. AI는 더 이상 이메일함에서 특정 표현만 찾지 않았습니다. 상황을 어떻게 이해했는지를 보여 주었습니다.

자료가 들어오는 대로 파일의 구조는 바뀌었습니다.

- 새로운 주제가 중요해지면 새 항목이 생겼습니다.
- 두 항목이 서로 다른 사항을 뒤섞기 시작하면 둘로 나뉘었습니다.

정보는 더 이상 쌓이기만 하지 않았습니다. 함께 일할 수 있는 모양을 갖춰 갔습니다.

그런데 이 첫 시스템은 정확하지만 충분하지 않은 답을 내놓았습니다.

공사는 끝났지만 그 경위는 사라졌다

공사가 끝나면 요약에는 이렇게 적혔습니다. “공사 완료, 검수 양호.”

이것은 정확한 현재 상태였습니다. 동시에 무언가를 잃은 결과였습니다.

- 공사는 언제 시작했나?
- 언제 지연됐나?
- 일정은 왜 바뀌었나?
- 변경 계약서는 언제 체결됐나?

이 모든 단계가 중요했습니다. 마지막 갱신은 최종 상태만 남기고 그 단계를 모두 지워 버렸습니다.

저는 요약을 요청했습니다. AI는 예전 상태를 새 상태로 바꾸며 요약을 최신 상태로 유지했습니다. 지시를 따랐지만, 그 지시는 경위를 보호하지 못했습니다.

그래서 같은 주제를 바라보는 두 가지 관점이 필요했습니다.

- 첫 번째는 “지금 어디까지 왔는가?”라는 질문에 빠르게 답해야 했습니다.
- 두 번째는 사건을 순서대로 보존해야 했습니다. 공사 시작, 지연, 새 합의, 작업, 검수입니다.
- 현재 상태는 행동을 가능하게 했습니다.
- 경위는 어떻게 이 상태에 이르렀는지 이해하고, 의심스러운 점을 검증하게 했습니다.

이 구분은 아주 구체적인 결함에서 태어났습니다. 저는 이를 정리 방식의 개선 정도로 생각했습니다. 그러나 이는 AI와 함께 일하는 제 방식에서 가

장 중요한 원칙 중 하나가 됐습니다. 지금 참인 것과 지금까지 일어난 일은 서로 다른 역할을 합니다.

AI는 제 규칙을 스스로 알 수 없었다

이름에서도 다른 문제가 드러났습니다.

- 두 업체 담당자의 성이 같았습니다.
  - 둘을 구분하려면 이메일의 주제를 이해해야 했습니다.
- 한 공동소유자는 때로는 결혼 전 성으로, 때로는 결혼 후 성으로 나타났습니다.

별도의 안내가 없으면 AI는 한 사람을 두 사람으로 나눌 수 있었습니다.

다른 차이는 업무 방식에서 나왔습니다.

- 100유로 미만을 미납한 사람은 다른 미납자와 같은 방식으로 관리하지 않았습니다.
- 이 기준은 AI가 알 수 있는 보편적인 규칙에서 나온 것이 아닙니다.
- 우리가 일하는 방식에서 나온 것입니다.

저는 이런 지식을 오래 사용할 규칙으로 정리해 파일에 명시적으로 적기 시작했습니다.

```
# 처리 규칙

- 지붕 문제에서 언급된 Martin은 지붕 업체를, 조경 문제에서 언급된
  Martin은 조경 업체를 뜻한다.
- Dupont-Lemoine과 Lemoine은 같은 공동소유자, 14호를 뜻한다.
- 100유로 미만의 미납자는 별도 목록에 표시한다.
```

그보다 더 복잡할 것은 없었습니다. 대화를 할 때마다 다시 말해야 했던 세 문장을 그 후로는 한 번도 반복할 필요가 없었습니다.

각 규칙은 실제로 관찰한 오류나 실제 모호함에서 나왔습니다. 제 머릿속에는 있지만 AI가 확신을 갖고 다시 구성할 수는 없는 지식을 담았습니다.

이로써 시스템은 서로 다른 두 가지 자료를 보존하기 시작했습니다.

- 공동주택에 관한 정보,

- 그 정보를 어떻게 처리하길 원하는지에 관한 정보입니다.

도움을 주던 파일이 방해가 되기 시작했다

몇 주가 흐르며 파일은 커졌습니다. 500줄을 넘고, 800줄을 넘고, 결국 1,000줄을 넘었습니다.

이런 성장은 파일이 유용했다는 증거였습니다.

- 주제를 더 잘 따라갈 수 있었고,
- 업체를 식별할 수 있었고,
- 결정을 다시 찾을 수 있었습니다.

하지만 시스템을 가능하게 한 바로 그 조건이 점점 사용을 제약하기 시작했습니다.

- 서로 다른 정보가 너무 많이 한 공간을 공유했습니다.
- 규칙과 사실이 뒤섞였습니다.
- 재정 정보 옆에 공사, 사람, 보험, 입찰 정보가 나란히 있었습니다.

올바른 자료를 찾는 데 다시 노력이 필요해졌습니다.

저는 먼저 처리 규칙을 따로 분리했습니다. 그 규칙은 공동주택을 설명하지 않았습니다. 공동주택에 관한 정보로 어떻게 일해야 하는지를 설명했습니다. 따로 놓으니 더 쉽게 찾고 적용할 수 있었습니다.

그다음에는 자료를 여러 주제로 나눴습니다. 공사는 재정, 업체, 총회의 결정, 보험과 마찬가지로 자기만의 공간을 갖게 됐습니다. 전체 개요는 각 공간에 무엇이 들어 있는지 알려 주었습니다. 그러면 AI는 질문을 받을 때마다 모든 자료를 훑지 않고, 해당 주제만 열 수 있었습니다.

마지막으로 법률 용어, 약어, 공동주택에서만 쓰는 표현 등 특수한 어휘를 한데 모았습니다. 백과사전을 만들려는 것은 아니었습니다. 대화를 이해하는 데 결정적인 단어에 안정적인 뜻을 부여하려는 것이었습니다.

제 머릿속에는 아직 거창한 아키텍처가 없었습니다. 문제가 하나의 구성요소를 필요로 할 때마다 그것을 하나씩 더했을 뿐입니다.

출처를 잃지 않고 신뢰하기

AI는 때로 실수했습니다. 정확히 같은 주제가 아닌 두 정보를 연결하거나, 성이 같은 두 사람 중 잘못된 업체 담당자를 고를 수 있었습니다. 그래서 시스템은 이해를 빠르게 도와야 했지만, 사실을 확인할 유일한 버전이 되어서는 안 됐습니다.

저는 원본 이메일을 모두 보관했습니다. 답을 확인해야 하거나 중요도가 높아지면 받은 원본 자료로 돌아갈 수 있었습니다. 요약은 빠르게 접근하게 했습니다. 출처는 확인하게 했습니다.

이 선택 덕분에 시스템을 실제로 사용할 수 있었습니다. 원본 자료에 접근할 수 있었기 때문에, 이메일을 정리하는 과정에서 이따금 생기는 부정확함을 받아들일 수 있었습니다. 은행 문서, 법규상 의무, 돌이킬 수 없는 결정에는 같은 수준의 신뢰가 적합하지 않을 것입니다. 위험은 항상 시스템에 무엇을 맡기는지, 그리고 확인이 가능한지에 달려 있습니다.

그래서 컨텍스트는 이메일함을 대체하지 않았습니다. 원본 메시지로 돌아갈 길을 남겨 둔 채, 이메일함의 내용을 작업 기억으로 바꾸었습니다.

이미 나타난 것

몇 주 만에 출발점이었던 파일 하나와 한 문장은 여러 필요에 답할 수 있는 집합이 됐습니다. 현재 상태를 보여 주고, 경위를 보존하고, 공동주택에 고유한 규칙을 적용하고, 주제를 구분하고, 용어를 안정시키고, 출처로 돌아갈 수 있었습니다.

새 이메일이 도착하면, 이전의 모든 것을 다시 구성하는 대신 그 이메일이 무엇을 바꾸는지에 집중할 수 있었습니다. 질문이 생기면 어떤 스레드에서 어떤 메시지를 찾아야 할지 고민하는 대신, 문제의 언어로 바로 물을 수 있었습니다. AI가 틀리면 규칙을 고치거나 원문으로 돌아갈 수 있었습니다.

저는 컨텍스트를 한 번에 설계하지 않았습니다. 요약은 경위를 지웠고, 이름은 서로 섞였고, 규칙은 사라졌고, 파일은 너무 커졌습니다. 저는 이런 부

족함이 이어지는 과정에서 그 시스템이 나타나는 모습을 지켜봤습니다. 각 수정은 작업을 더 오래 남게 하고 더 쉽게 이어서 할 수 있게 했습니다.

이 시작은 중요합니다. 컨텍스트는 더 많이 분류하고 싶은 욕구에서 태어난 것이 아닙니다. 변하는 자료를 이해하고, 그것을 확인할 수 있게 하는 것을 보존하며, 처음부터 다시 시작하지 않고 일을 계속할 필요에서 태어났습니다.

기억할 점 – 같은 주제를 보는 두 가지 관점

현재 상태는 “지금 어디까지 왔는가?”에 답합니다. 경위는 “어떻게 여기까지 왔는가?”에 답합니다. 이전 요약을 덮어쓰며 최신 상태만 남기면 첫 번째는 보존하지만 두 번째는 잃습니다. 둘 다 필요합니다. 하나는 행동하기 위해, 다른 하나는 이해하고 확인하기 위해 필요합니다.

1.2

첫 번째 정의

500개의 이메일에서 태어난 이 시스템에 이제 첫 번째 정의를 내릴 수 있습니다.

> 컨텍스트는 하나의 주제에 전념하는 작업 공간입니다. 사람과 *AI*는 이곳에 오랜 기간 이해하고, 결정하고, 결과물을 만드는 데 필요한 것을 보존합니다.

이 정의는 의도적으로 간단합니다. 이 책에서 다루는 컨텍스트에는 명확한 구조가 있으며, 책의 후반부에서 자세히 설명합니다. 처음에는 그 가치를 이해하기 위한 여섯 가지 핵심 요소만 있으면 됩니다.

- 목적,
- 자료,
- 아이디어,
- 작업 상태,
- 작업에 고유한 규칙,
- 만들어야 할 결과물입니다.

하나의 주제와 목적

공동주택 컨텍스트는 제가 아는 모든 것을 담으려 하지 않았습니다. 현재 진행 중인 사안을 이해하고, 그 경위를 보존하며, 공동소유자 협의회에서 행동하기 위한 정보를 찾는 목적이 있었습니다.

이 목적은 경계를 만들었습니다. 지붕에 관한 이메일은 그 안에 들어갔습니다. 공동주택과 관계없는 흥미로운 글은 들어갈 자리가 없었습니다. 이 한계 덕분에 시스템은 흥미롭다고 판단된 모든 정보가 쌓이는 창고가 되지 않았습니다.

따라서 컨텍스트는 하나의 의도에서 시작합니다. 다음과 같은 일에 쓸 수 있습니다.

- 책 쓰기,
- 제품 문제 탐구하기,
- 결정 준비하기,
- 제안요청서에 답하기,
- 업무 대상 관리하기,
- …

출처를 보존하는 자료

자료는 두 가지 통로로 컨텍스트에 들어옵니다.

첫 번째 통로는 작업 세션 전부터 존재했던 것을 받아들입니다.

- 이메일,
- 글,
- 책,
- 웹 페이지,
- 인터뷰 녹취록
- …

이것이 문서 출처입니다. 이들은 이미 만들어진 대상으로 들어와 안정적으로 남습니다. 공동주택 사례에서 작업 연기를 알린 이메일은 첫 번째 분류에 속합니다.

두 번째 통로는 AI와 함께 일하는 동안 표현된 것을 받아들입니다. 제가 상황을 설명하고, 해석을 바로잡고, 미묘한 차이를 부연하거나, 경험에서 얻은 확신을 말할 때 그 말은 세션 출처가 됩니다. 이는 대화가 진행되는 동안 점진적으로 만들어집니다. 두 이름이 같은 공동소유자를 가리킨다고 제가 설명하면, 그 말은 먼저 두 번째 통로로 들어옵니다.

차이는 자료가 언제, 어떤 방식으로 나타나는지에 있습니다.

- 문서는 세션 전에 존재하며 세션으로 가져옵니다.
- 세션 출처는 세션 중에 태어나 대화와 함께 자랍니다.

이후 두 출처는 같은 기능을 합니다. 해석하기 전의 자료를 보존합니다.
- 그 안의 모든 내용이 사실임을 보장하지는 않습니다.
- 받았거나 표현된 내용을 다시 찾고, 그 신뢰성을 판단할 수 있게 합니다.

컨텍스트는 그런 자료에서 아이디어, 규칙, 작업 상태를 끌어낼 수 있습니다. 변환된 지식은 다른 자리를 차지합니다. 날짜가 틀린 것 같거나, 두 사람을 헷갈렸거나, 해석에 의문이 생기면 그 출처로 돌아갈 수 있습니다.

사용할 수 있게 만든 아이디어

출처를 모으는 것만으로는 충분하지 않습니다. 잘 보존된 500개의 이메일은 여전히 다시 읽어야 하는 500개의 이메일입니다.

그래서 컨텍스트는 이 자료를 점진적으로 변환합니다. 다음을 드러냅니다.
- 주제,
- 중요한 사실,
- 규칙,
- 구분,
- 미해결 질문,
- 관계입니다.

따로 떨어져 있던 하나의 대화는 그것이 속한 공사의 경위와 합쳐집니다. 같은 업체에 대한 여러 언급은 하나의 공통 지식을 풍부하게 만듭니다. 한 번의 정정은 같은 혼동이 반복되지 않게 합니다.

이 중간 단계가 중요한 이유는 정보를 사용할 수 있게 만들기 때문입니다.
- 받은 자료는 흔적을 보존합니다.
- 정리된 지식은 이해와 행동을 가능하게 합니다.

더 복잡한 작업에서는 일부 아이디어를 여러 번 쓸 수 있습니다. 같은 제품 지식이 결정, 명세서, 문서화에 활용됩니다. 검증된 하나의 사실은 여러 고객 질문지에 답하는 데 도움이 됩니다. 결과물은 바뀌어도 아이디어와 그 출처는 계속 사용할 수 있습니다.

계속할 수 있게 하는 상태

진행 중인 작업에는 언제나 '지금'이 있습니다. 우리가 알고 있는 것, 아직 확실하지 않은 것, 이미 결정한 것, 그리고 다음에 일어나야 할 것입니다.

공동주택 사례에서 상태는 즉각적인 질문에 답했습니다. 이 공사는 오늘 어디까지 진행됐는가? 그 경위는 다른 질문에 답했습니다. 어떻게 이 지점까지 왔는가? 실제로 작업을 이어서 하려면 시스템에는 둘 다 필요했습니다.

이 구분은 공사 진행 관리를 넘어서도 적용됩니다. 글을 쓰다가 중단하거나 제품 문제를 탐구하다가 멈추면, 저는 문서 묶음 이상을 다시 찾고 싶습니다. 다음을 알고 싶습니다.

- 확정된 것,
- 아직 취약한 지점,
- 작업을 다시 움직이게 할 수 있는 다음 행동입니다.

컨텍스트는 대화 밖에 이 연속성을 보존합니다.

- AI와의 세션은 멈출 수 있습니다.
- 작업은 상태를 유지하고, 다음 세션은 그 상태로 돌아갈 수 있습니다.

작업에 고유한 규칙

AI에게는 일반적인 지식이 있습니다. 그러나 특정한 상황을 특별하게 만드는 많은 것은 모릅니다.

- 같은 성을 쓰는 두 업체 담당자,
- 팀이 고객 신호를 분류하는 방식,
- 주장을 공개하기 전에 기대하는 근거 수준
- 책에서 선택한 문체,
- …

컨텍스트는 이 규칙을 그 규칙이 적용되는 작업과 함께 보존합니다. 경험을 통해 한계가 드러나면 고칠 수 있습니다. 그 뒤로는 사람이 요청할 때마다 다시 표현하지 않아도 적용됩니다.

이 규칙이 AI에게 작업의 책임을 넘기는 것은 아닙니다. 사람이 AI에게 무엇을 기대하는지 명시할 뿐입니다. 사람은 목적을 정하고, 자료를 제공하거나 검증하고, 해석을 바로잡고, 결정합니다. AI는 이 틀 안에서 흡수하고, 연결하고, 정리하고, 질문하고, 다시 표현하며, 결과물을 준비합니다.

필요에 맞춘 결과물

컨텍스트는 해야 할 일을 지향합니다.

공동주택에서 결과물은 다음과 같을 수 있습니다.

- 질문에 대한 답,
- 공사 현황,
- 토론 준비입니다.

다른 활용에서는 다음과 같을 것입니다.

- 글,
- 제안요청서,
- 문서화입니다.

같은 공간에서 여러 결과물을 만들 수 있는 이유는 자료를 최종 형태로 바꾸기 전에 보존하기 때문입니다. 출처는 처음 사용된 문서 안에 갇히지 않습니다. 아이디어는 다른 질문이나 다른 독자를 위해 계속 사용할 수 있습니다.

이 능력은 컨텍스트를 단순한 아카이브와 구분합니다.

- 아카이브는 보존할 수 있게 합니다.
- 컨텍스트는 보존한 것을 다시 꺼내 이어 가고 변환하는 일을 조직합니다.

아주 간단하게 시작할 수 있는 형태

제가 사용하는 컨텍스트는 제 작업을 돕는 도구와 독립적으로 읽고, 옮기고, 보존할 수 있는 파일로 이루어져 있습니다. 이 형태는 통제권과 지속성을 원하는 제 필요에 맞춥니다. 같은 기능을 유지할 수 있다면 다른 구현 방식도 가능합니다.

그러나 첫 번째 컨텍스트에는 복잡한 것이 하나도 없었습니다. 파일 하나에 담겨 있었습니다. 한계가 눈에 보일 때마다 자랐습니다. 이 점진적인 과정은 여전히 중요합니다. 이 책에서 설명하는 시스템을 처음부터 모두 구축해야만 아이디어를 시험해 볼 수 있는 것은 아닙니다.

하나의 주제, 하나의 목적, 첫 번째 자료면 충분합니다.

- AI가 이를 정리하도록 돕습니다.
- 사람은 무엇이 부족한지 살펴보고, 규칙을 추가하거나, 출처를 보존하거나, 역할이 다른 두 정보를 분리합니다.
- 마주하는 문제에 맞춰 구조가 나타납니다.

그러므로 컨텍스트는 특정한 폴더 구조라기보다, 사람과 AI, 그리고 목적을 향한 자료 사이의 오래 지속되는 작업 관계입니다.

다음 부에서는 이 관계가 무엇을 바꾸는지 살펴봅니다. AI와의 대화만으로도 이미 풍부한 결과를 낼 수 있습니다. 그 대화가 현재의 순간을 넘어 목적과 경위, 상태를 보존할 수 있는 환경에 담기면 컨텍스트로의 전환이 시작됩니다.

기억할 점 – 컨텍스트의 여섯 가지 핵심 요소

경계를 그어 주는 목적, 출처를 보존하는 자료, 사용할 수 있게 만든 아이디어, 계속할 수 있게 하는 상태, 작업에 고유한 규칙, 만들어야 할 결과물입니다. 이 여섯 가지면 시작하기에 충분합니다. 구조는 이후 문제가 필요로 할 때 나타납니다.

부 2

컨텍스트가 바꾸는 것

AI와의 대화만으로도 이미 많은 것이 만들어집니다. AI는 설명하고, 다시 표현하고, 제안하며, 종종 혼자서 찾았을 답보다 나은 답을 줍니다.

그래서 이 부에서는 컨텍스트가 여기에 무엇을 더하는지 살펴봅니다. 답은 얻은 답변의 품질이 아니라 두 세션 사이에 무엇이 남는지에 있습니다. 작업은 멈춘 지점에서 다시 시작되고, 문서들은 서로 어긋나는 내용을 매끄럽게 덮지 않고 그대로 보존합니다.

2.1

대화에서 다시 이어갈 수 있는 작업으로

AI와의 대화는 아주 간단하게 시작할 수 있습니다.

아직 정리되지 않은 아이디어를 AI에게 줍니다. AI가 다시 표현합니다. 구성을 하나 요청하고, 다시 다른 구성을 요청합니다. 제가 생각하지 못한 반론을 AI가 제안합니다. 저는 의도를 더 명확히 하고, 한 방향을 거절하고, 다른 방향을 더 깊게 파고듭니다. 몇 분 만에 대화는 오랫동안 빈 페이지 앞에 머물렀을지도 모를 작업을 앞으로 나아가게 합니다.

질문 하나면 시작할 수 있다는 것은 채팅의 큰 장점 중 하나입니다. 이 형식은 짧은 작업에 매우 잘 맞습니다. 요약이나 설명, 몇 가지 가능성을 요청합니다. 답을 받아 쓰고 나면 대화는 끝나도 됩니다.

작업이 오래 걸리면 관계가 바뀌어 갑니다.

글을 쓰기 전에 필요한 다섯 문장

지난주에 쓰기 시작한 글을 다시 이어 가기 위해 6일 전의 대화를 다시 엽니다. 한 줄을 쓰기도 전에 저는 다음과 같이 입력합니다.

> 작업 보고서의 품질에 관한 글을 다시 이어서 작성해 줘. 독자는 개발자가 아니라 운영 책임자야. 폼을 짧게 만드는 방안은 제외했어. 아직 결론을 내리기에는 일러. 감사에 관한 수치는 인터뷰 하나에서만 나왔으니 가설로 다뤄 줘. 우리는 세 번째 섹션까지 왔어.

6일 전에 AI가 이미 보고 있었던 작업으로 AI를 다시 데려오려고 다섯 문장을 입력한 것입니다.

지금 입력한 모든 내용은 제가 다시 입력하기 전부터 존재했습니다. 관점, 독자, 결정, 취약점, 글의 현재 위치는 이전 대화에 담겨 있었습니다. 대화가 그 정보를 말한 순서대로 놓아두었기 때문에, 저는 제 기억에서 그것을 꺼내어 다시 적었을 뿐입니다.

하지만 작업은 다른 방식으로 조직됩니다.

- 처음에 내린 지시는 끝까지 계속 적용되어야 합니다.
- 어제 추가한 출처는 내일 쓸 단락의 근거가 될 수 있어야 합니다.
- 이전에 내린 결정은 같은 논의를 다시 여는 일을 막아야 합니다.
- 대기 중인 사항은 적절한 때에 다시 나타나야 합니다.

대화가 길어질수록 사람은 이 구조를 머릿속에 더 많이 유지해야 합니다. 새 세션이 시작되면 사람이 AI를 위해 상황을 다시 구성합니다. 이것이 목적이고, 이것이 바뀌었고, 이것이 문체이고, 이것이 결정이고, 여기까지 왔다고 설명합니다.

채팅은 여전히 유용합니다. 그러나 연속성을 지키는 부담은 채팅을 사용하는 사람에게 있습니다.

오래 남아야 하는 것에 자리를 마련하기

컨텍스트는 작업에 대화와 분리된 공간을 부여함으로써 이 관계를 바꿉니다.

대화는 여전히 의견을 주고받는 곳입니다. 질문하고, 탐구하고, 명확히 하고, 바로잡을 수 있습니다. 오래 남아야 할 것은 작업 공간으로 옮겨 갑니다.

- 목적,
- 출처,
- 규칙,
- 안정된 아이디어,
- 결정,
- 글,
- 진행 상태입니다.

차이는 작아 보일 수 있습니다. 회의와 그 회의가 다루는 프로젝트 사이의 차이와 비슷합니다. 좋은 회의는 프로젝트를 앞으로 나아가게 하지만, 프로젝트 전체가 회의에 대한 기억 안에만 담기지는 않습니다. 문서, 결정, 행동은 회의가 끝난 뒤에도 계속 존재합니다.

채팅 전체 녹취록을 보관해도 같은 효과를 내지는 못합니다. 무슨 말을 했는지 정확히 다시 찾을 때는 가치가 있지만, 녹취록은 대화의 형태를 따릅니다. 질문, 답, 정정, 샛길로 빠지기, 주제로 돌아오기의 순서입니다. 작업을 다시 이어 가려면 이 연속된 대화를 다시 읽고 지금 중요한 것을 판단해야 합니다.

반면 컨텍스트는 대화에서 나온 내용을 변환합니다.

- 정정 내용은 동일한 오류의 재발을 막을 수 있는 자리로 갑니다.
- 결정은 결정으로서 계속 눈에 보입니다.
- 출처는 그것이 뒷받침하는 내용과 연결된 채 남습니다.
- 글은 가장 최신의 완전한 버전으로 존재합니다.

이렇게 기억은 행동을 향합니다. 과거를 보존하는 데만 머물지 않습니다. 다음 작업을 준비합니다.

32개의 글을 다시 찾게 해 주는 한 페이지

이 책을 다시 작업하며 저는 이 차이가 만드는 변화를 확인했습니다.

그때 원고에는 32개의 글이 있었습니다. 일부는 쓰여 있었고 나머지는 아직 비어 있었습니다. 한 섹션은 비중이 너무 컸기 때문에 둘로 나뉘어 있었습니다. 두 대목은 제 검토를 기다리고 있었습니다. 한 주장은 실제 사례에 근거하지 않아 다시 작성해야 했습니다.

이 요소들을 하나씩 보면 어렵지 않습니다. 그러나 한데 모으면 유용한 한 줄을 쓰기 전에 다시 구성해야 할 상황이 됩니다.

32개의 글을 열고, 구성을 다시 읽고, 예전 대화를 훑어보며, 대기 중인 메모를 찾을 수도 있습니다. 이런 고고학적 작업은 시간을 쓰게 하고 의심을 남깁니다. 바로 오류를 막아야 했던 그 사항을 놓친 것은 아닐까?

대신 저는 책의 컨텍스트를 엽니다. 그날 아침, 컨텍스트는 다음과 비슷한 내용의 재개 파일을 내밀었습니다.

```
# 작업 재개

## 현재 상태
구성부터 결론까지 32개의 글이 존재한다.
기술 부분은 작성됐지만, 첫 두 부분은 아직 너무 이론적이다.

## 계속하기 전에 알아야 할 것
예상 독자는 이 주제를 처음 접한다. 실제 사례보다 개념이 먼저 나오면 따라오지 못한다.
문서 시스템의 실패에 관한 주장은 아직 개인적인 경험에 근거하지 않았다.

## 다음 행동
공동주택 관련 이메일 500통 사례를 중심으로 도입부를 다시 쓴다.

## 아직 열려 있는 사항
두 대목이 내 검토를 기다리고 있다.
여러 출처가 노트로 변환되지 않은 채 들어와 있다.
```


1분도 걸리지 않아 읽을 수 있는 한 페이지입니다. 저는 무엇을 해야 하는지 알고, 32개의 글 중 하나도 열지 않았습니다.

계속하기와 재개하기는 다르다

계속하기는 다음 행동을 아는 것입니다. 재개하기는 그 행동을 하기 전에 작업의 의미를 다시 찾는 것입니다.

할 일 목록은 출처 하나를 처리해야 한다고 알려 줄 수 있습니다. 그러나 왜 그 출처가 우선인지, 책의 어느 부분을 밝혀야 하는지, 어떤 위험을 바로잡는 데 쓰여야 하는지는 설명하지 않습니다.

재개하려면 더 풍부한 정보가 필요합니다.

- 작업이 추구하는 목적,
- 작업이 실제로 도달한 지점,
- 앞으로의 방향에 여전히 영향을 미치는 결정,
- 잃어서는 안 되는 취약점,
- 다시 시작할 수 있을 만큼 구체적인 다음 행동입니다.

재개 파일도 바로 이 순서로 읽힙니다. “32개의 글”은 규모를 알려 줍니다. “실제 사례보다 개념이 먼저 나오면 독자가 따라오지 못한다”는 방향을 알려 줍니다. 다음 행동은 두 정보에서 나오며, 책을 열 구체적인 사례를 지정합니다.

재개는 목록의 위치만이 아니라 하나의 추론을 다시 구성했습니다.

그렇다고 재개 파일이 지난 세션의 모든 세부 사항을 제 머릿속에 되살리려 하지는 않습니다. 너무 짧은 요약은 선택의 이유를 지웁니다. 너무 긴 보고서는 작업을 다시 시작하는 순간을 늦춥니다.

중단은 시점을 고르지 않는다

처음에 저는 재개를 깔끔한 마무리의 결과로 생각했습니다. 하나의 세션이 끝나면 상태를 정성껏 요약하고 다음 행동을 적어 둘 것이라고 생각했습니다.

현실은 그런 각본을 거의 따르지 않습니다.

- 회의가 시작됩니다.
- 배터리가 방전됩니다.
- 하루가 예상보다 빨리 끝납니다.
- 다른 긴급한 일이 주제의 자리를 차지합니다.

재개를 위한 기억이 가장 유용한 시점은 바로 마무리를 준비하지 못한 시점입니다.

그래서 컨텍스트는 작업하는 동안 재개 능력을 유지합니다. 결정은 내릴 때, 할 일은 표현할 때, 출처는 들어올 때 각자의 자리를 찾습니다.

이렇게 재개는 사용자가 영웅적으로 지켜야 하는 습관이 아니라 시스템의 속성이 됩니다.

그래도 사람에게는 결정적인 역할이 남습니다. 제안된 다음 행동이 더 이상 우선이 아니라고 판단할 수 있고, 예전 결정을 다시 검토해야 한다고 판단할 수 있으며, 새로운 사실이 목적을 바꾸었다고 판단할 수 있습니다. 기억은 경로를 고정하지 않습니다. 경로를 바꾸기로 결정하는 출발점을 눈에 보이게 합니다.

대화를 다시 열 때마다 다시 입력하던 다섯 문장을 저는 더 이상 입력하지 않습니다. 그 문장들은 어딘가에 적혀 있고, 매 세션을 시작할 때 저 대신 다른 누군가가 읽습니다.

기억할 점 – 채팅, 작업 환경, 재개

채팅은 의견을 주고받는 곳이고, 컨텍스트는 오래 남아야 할 것의 자리입니다. 대화는 정보를 말한 순서대로 배치하지만, 작업에는 다른 질서가 필요합니다. 계속하기는 다음 행동을 아는 것이고, 재개하기는 그 행동을 하기 전에 작업의 의미를 다시 찾는 것입니다. 중단은 예고하지 않습니다. 재개 능력은 닫을 때가 아니라 작업하는 동안 유지합니다.

2.2

두 고객, 하나의 백서, 그리고 보존해야 할 모순

지금껏 컨텍스트와 그것이 만드는 여러 효과를 설명했습니다. 그러나 실제로 열어 보면 어떤 모습일까요? 파일 안에서는 무엇을 보게 될까요? 서로 다른 여러 출처는 각자는 옳지만 함께 받아들이기 어려운 아이디어를 어떻게 드러낼까요?

이 가상 컨텍스트의 주제는 간단합니다. 제품을 바꿀지 결정하기 전에 일부 기술자가 왜 불완전한 작업 보고서를 작성하는지 이해하는 것입니다.

임무는 몇 줄에 담깁니다.

> 작업 보고서의 품질을 떨어뜨리는 요인을 이해한다. 입력의 어려움, 조직적 제약, 작업 데이터의 활용 필요를 구분한다. 여러 출처를 비교하기 전에 해법을 선택하지 않는다.

이 임무는 작업에 방향을 부여합니다. 아직 기능을 설계하는 것이 아니라, 문제를 이해하라고 요구합니다.

첫 번째 피드백에서 하나의 아이디어가 나타난다

컨텍스트에 처음 들어온 문서는 운영 책임자와의 인터뷰 녹취록입니다.

그는 다음과 같이 설명합니다.

> "기술자들은 금요일에 작업 보고서를 몰아서 작성하는 경우가 많습니다. 현장에서는 폼이 너무 길다고 느끼거든요. 주말이 다가오면 세부 내용의 일부를 이미 잊어버립니다."

AI는 이 인터뷰를 출처로 보존합니다. 그런 다음 별도의 카드에 하나의 아이디어를 추출합니다. 저는 이것을 원자 노트라고 부릅니다.

> 입력 요구가 너무 까다로우면 일부 기술자가 보고서 작성을 미루고 세부 내용을 잃을 수 있다.

이 문장은 모든 기술자가 그렇게 행동한다고도, 폼의 길이가 유일한 원인이라고도 말하지 않습니다. 증언의 핵심을 유지합니다. 묘사된 조직에서는 입력이 미뤄지고, 작업과 글 작성 사이에 정보의 일부가 사라집니다.

출처는 계속 사용할 수 있습니다. 노트는 출처를 대체하지 않습니다. 주요 아이디어를 더 쉽게 다시 찾고, 이후의 자료와 비교할 수 있게 합니다.

두 번째 피드백이 같은 노트를 풍부하게 만든다

몇 주 뒤, 두 번째 인터뷰가 컨텍스트에 들어옵니다.

이번에는 다른 운영 책임자가 다음과 같이 말합니다.

> "데이터 품질을 높이려고 *12*개 항목을 필수로 만들었습니다. 그랬더니 일부 기술자는 제출하기 위해 '특이 사항 없음'이라고 쓰거나 하이픈 하나만 입력합니다."

이 증언은 정확히 같은 행동을 묘사하지는 않습니다. 첫 번째 인터뷰의 기술자는 입력을 미룹니다. 두 번째 인터뷰의 기술자는 폼을 작성하지만 요구사항을 우회합니다.

그래도 두 출처는 같은 핵심 아이디어를 다룹니다. 더 많은 정보를 얻으려고 만든 제약이 실제 입력 정보의 품질을 떨어뜨릴 수 있다는 것입니다.

그래서 두 번째 인터뷰는 첫 번째 원자 노트에 합쳐집니다. 첫 번째 출처를 대체하지도, 경쟁하는 아이디어를 만들지도 않습니다. 지연 작성에 이어 겉으로만 완료한 입력이라는, 같은 문제의 다른 형태를 드러내며 노트를 풍부하게 만듭니다.

이제 노트는 다음과 같이 말할 수 있습니다.

> 입력 요구가 너무 까다로우면 일부 기술자가 보고서 작성을 미루거나 요구를 우회해, 수집된 정보의 품질을 떨어뜨릴 수 있다.

두 출처가 참조됩니다. 각자의 역할과 한계는 그대로 남습니다. 두 증언만으로 현상의 규모를 측정할 수 있다고 주장하지 않으면서도 노트는 더 단단해집니다.

백서에서 다른 아이디어가 나타난다

그다음에 작업 데이터 활용에 관한 백서를 추가합니다.

이 문서는 다른 필요를 주장합니다. 반복되는 고장을 이해하고, 설비를 비교하고, 특정 감사를 준비하려면 기업에는 충분히 구조화된 보고서가 필요합니다. 기술자마다 자유 입력란에 원하는 내용을 쓰거나 아무것도 입력하지 않으면, 작성은 쉬워질지 몰라도 데이터는 조직 전체에서 활용할 가치를 크게 잃습니다.

이 출처가 앞선 증언이 틀렸다는 뜻은 아닙니다. 앞서 묘사한 기술자는 여전히 입력을 미루거나 우회합니다. 이 PDF는 다른 현실을 드러냅니다. 모든 제약을 없애는 것에도 비용이 따른다는 것입니다.

AI는 이 요소를 첫 번째 노트에 합친 뒤, 일반적인 절충안으로 다시 쓰고 싶어 할 수 있습니다. 그렇게 하면 안 됩니다. 백서는 인터뷰에서 관찰한 사실에 단서를 더하지 않습니다. 다른 주장을 제시합니다.

그래서 백서는 두 번째 원자 노트를 만듭니다.

> 작업을 비교하고 작업 데이터를 활용하려면 구조화된 보고서가 필요하다.

컨텍스트에는 이제 서로 다른 두 개의 아이디어가 있습니다.

- 입력 제약은 실제로 수집되는 정보의 품질을 떨어뜨릴 수 있습니다.
- 구조가 없으면 이후에 그 정보를 활용하는 품질이 떨어질 수 있습니다.

두 노트는 동시에 참일 수 있습니다. 그럼에도 제품 결정을 서로 반대 방향으로 끌어당깁니다.

노트 사이의 대립은 눈에 보이게 남아야 한다

컨텍스트의 역할은 모순이 사라질 때까지 첫 번째 노트를 고치는 것이 아닙니다. 두 주장을 모두 보존하고, 서로의 관계를 명확히 드러내야 합니다.

이때 주제 노트는 두 아이디어를 함께 살펴볼 때 알 수 있는 것을 다음과 같이 표현할 수 있습니다.

> 작업 보고서의 품질은 입력할 때 요구되는 노력과, 이후의 활용에 필요한 구조 사이의 긴장에 달려 있다.

이 세 번째 노트는 외교적인 타협안이 아닙니다. 두 원자 노트 중 어느 것도 대체하지 않습니다. 둘을 함께 읽을 때 새로 드러나는 문제를 보여 줍니다.

모순은 작업 상태에도 반영됩니다. 입력을 미루거나 우회하게 만들지 않으면서 어떻게 구조화된 정보를 얻을 것인가? 세 출처 중 어느 것도 아직 답을 주지 못합니다.

화면에 보이는 것

이 예시에 쓴 컨텍스트는 작은 파일 폴더 형태로 존재합니다. 이 구조 자체가 개념인 것은 아닙니다. 이는 현재 제가 개념에 구체적인 형태를 부여하는 방식입니다.

```
contexte-qualite-comptes-rendus/
├── 00-frame/
│   ├── mission.md
│   └── directives.md
├── 01-state/
│   ├── state.md
│   └── contradictions.md
├── 03-sources/
│   ├── SRC00001-entretien-exploitation.md
│   ├── SRC00002-entretien-exploitation.md
│   └── SRC00003-livre-blanc-donnees-intervention.md
├── 04-notes/
│   ├── atomic/
│   │   ├── ATO00001-la-contrainte-de-saisie.md
│   │   └── ATO00002-des-comptes-rendus-structures.md
│   └── thematic/
│       └── THM00001-la-qualite-depend-de-la-saisie.md
└── 05-outputs/
    └── questions-prochain-entretien.md
```

이제 책 처음부터 사용한 단어를 눈에 보이는 대상과 연결할 수 있습니다.

- 임무는 폴더가 왜 존재하는지 설명합니다.
- 출처는 해석하기 전에 받은 것을 보존합니다.
- 원자 노트는 서로 다른 아이디어를 안정적으로 담습니다.
- 주제 노트는 이들 사이의 긴장을 이해할 수 있게 합니다.
- 상태는 이 긴장이 어떤 문제를 열어 두었는지 말합니다.
- 결과물은 다음 행동을 준비합니다.

이 모든 것에 대단한 비밀 소프트웨어가 필요한 것은 아닙니다. 여기서 VS Code의 창을 통해 컨텍스트에 접근합니다. Codex 앱이나 Claude Cowork이어도 됩니다. 컨텍스트 자체는 파일, 각 파일의 기능, 그리고 파일 사이에 보존된 관계에 있습니다.

피드백에 맥락이 생긴다

며칠 뒤 저는 작업을 다시 시작하며 AI에게 제안합니다.

> “모든 항목을 선택 사항으로 만드는 것은 어떨까?”

독립된 대화에서라면 AI는 이 아이디어를 지지하거나, 더 간단한 인터페이스를 제안하거나, 장점을 목록으로 만들 수 있습니다. 컨텍스트 안에서는 두 원자 노트와 두 노트를 잇는 긴장을 만납니다.

그러면 다음과 같이 답할 수 있습니다.

> 이 제안은 인터뷰에서 묘사한 마찰의 일부를 줄일 가능성이 있습니다. 그러나 백서에서 지적한 구조화된 데이터의 필요와는 충돌합니다. 출처는 두 제약이 모두 실제라는 것을 보여 주지만, 아직 단순화, 단계적 입력, 음성 입력, 작업 유형에 따른 항목 조정 사이에서 선택할 정보는 주지 못합니다.

이 피드백은 일반적인 프로덕트 매니지먼트 지식에서 나오지 않습니다. 현재 작업 안에 자리 잡고 있습니다. AI는 우리가 이해하려는 질문, 어떤 출처가 어떤 아이디어를 뒷받침하는지, 어떤 모순이 아직 열려 있는지 알고 있습니다.

AI가 제 대신 결정하지는 않습니다. 새로운 직감이 이미 확인된 현실 중 하나를 지우지 못하게 할 뿐입니다.

첫 번째 결과물이 다음 출처를 준비한다

컨텍스트의 상태는 "새 폼을 설계한다"고 말하지 않습니다. 확실해 보이는 것과 아직 불확실한 것을 구분합니다.

> 확실해 보이는 것 입력 제약은 작성 지연과 우회를 일으킬 수 있습니다. 그러나 보고서를 활용하려면 충분한 구조가 여전히 필요합니다.
>
> 아직 열려 있는 것 아직 어떤 마찰이 가장 큰 부담이 되는지, 어떤 정보가 반드시 구조화된 형태로 남아야 하는지 모릅니다.
>
> 다음 행동 해법을 바로 제시하지 않고 문제의 원인을 구분하는 인터뷰를 준비합니다.

그래서 첫 번째 결과물은 소박합니다. 새 인터뷰에서 쓸 질문 가이드입니다. 작성 시점, 현장의 제약, 각 항목의 실제 쓸모, 요구사항을 우회하는 방식을 이해하는 데 쓰여야 합니다.

결과물은 모순에 대해 결론을 내리지 않습니다. 나중에 결론을 내릴 수 있게 해 줄지도 모를 자료를 준비합니다.

모든 것을 합치는 기계가 아닌 작업 대상

이 예시는 AI가 범할 수 있는 오류를 또 하나 보여 줍니다. 주제가 비슷하다는 이유로 새로운 출처를 기존 노트에 억지로 넣으려는 오류입니다.

여러 출처가 같은 주장을 다룰 때 원자 노트는 성장할 수 있습니다. 두 인터뷰가 그런 경우입니다. 그러나 백서는 그 주장을 뒷받침하는 새로운 근거가 아닙니다. 백서에서는 자체 노트를 만들 가치가 있는 다른 아이디어가 나타납니다.

이 구분은 중요합니다. 대립하는 내용이 나올 때마다 이전 노트를 다시 쓰면 '아이디어의 세계'는 결국 긴장을 평탄하게 만듭니다. 지식은 일관되어 보이지만, 출처가 실제로 제공한 내용과는 더 이상 맞지 않습니다.

컨텍스트는 AI가 항상 올바른 경계를 선택하도록 보장하지 않습니다. 사람은 합치기를 거부하고, 새 노트를 요청하고, 노트 사이에 만든 관계를 확인할 수 있어야 합니다. 구조는 이 수정을 가능하고 지속적으로 만듭니다. 자동으로 결정해 주지는 않습니다.

이제 작업 대상을 직접 보았습니다.

- 손대지 않고 남아 있는 출처,
- 각자의 핵심을 지키는 원자 노트,
- 합치지 않고 관계를 맺는 주제 노트,
- 아직 열려 있는 모순과 이후의 작업을 준비하는 결과물입니다.

다음 장들에서는 컨텍스트를 여러 방식으로 활용한 사례를 돌아본 뒤, 이제 모습을 드러내기 시작한 이 요소를 바탕으로 그 구조를 더 자세히 열어 볼 것입니다.

기억할 점 – 모순은 보존한다

두 출처는 동시에 참이면서 결정을 서로 반대 방향으로 끌어당길 수 있습니다. 새 출처가 같은 주장을 강화할 때 원자 노트는 성장합니다. 다른 주장은 자체 노트를 가져야 합니다. 긴장을 없애려고 합치면 출처와 더 이상 일치하지 않는, 겉으로만 일관된 지식이 만들어집니다.

부 3

세 가지 구체적인 활용법

앞의 두 부에서는 컨텍스트가 무엇을 바꾸는지 설명했습니다. 이번에는 제가 실제로 사용하는 세 가지 상황에서 작동하는 모습을 보여 줍니다.

제안요청서에 답하기, 아직 모양을 갖추지 못한 생각으로 글을 쓰기, 잘게 끊기는 하루 속에서 제품 주제를 발전시키기입니다. 서로 다른 세 가지 기간, 자료, 독자를 다루지만, 매번 같은 흐름을 따릅니다. 이미 이해한 것을 다시 찾고, 변환하고, 그 과정에서 쓴 지식을 소진시키지 않으면서 결과물을 만듭니다.

3.1

이미 해 놓은 일을 잃지 않고 제안요청서에 답하기

이 시스템으로 처음 처리한 제안요청서에는 마흔세 개의 질문이 있었습니다.

제품 아키텍처, 보안, 백업, 서비스 연속성, 데이터 보호에 관해 답해야 했습니다. 일부 정보는 사내 문서에 있었습니다. 다른 정보는 설정 파일 깊숙이 묻혀 있었습니다. 어떤 정보는 서비스를 운영하는 팀에 물어봐야 했습니다.

기술 자료 일곱 개를 모으고 담당 팀과 한 차례 의견을 주고받아야 했습니다. 여기에 답변을 쓰는 일 자체도 더해졌습니다. 잠재 고객이 무엇을 묻는지 이해하고, 정보를 찾아내고, 그 정보가 여전히 사실인지 확인하고, 어디까지 공개할 수 있는지 결정한 뒤, 요구된 형식에 맞춰 답을 작성해야 했습니다.

특별한 상황은 아니었습니다. 새로운 잠재 고객이 올 때마다 저마다 다른 문서를 보내왔습니다. 어떤 때는 스프레드시트였고, 어떤 때는 워드 파일이나 PDF였습니다. 참조 기준도 문장도 달라졌지만, 묻는 주제는 늘 비슷했습니다.

그런데도 우리는 매번 처음부터 다시 시작하는 기분이었습니다. 답변에는 세 시간에서 여섯 시간이 걸렸고 여러 사람이 동원됐지만, 그 잠재 고객이 실제 고객이 된다는 보장은 없었습니다. 문서를 보내고 나면, 들인 노력은 발송한 파일 안에 갇힌 채로 남았습니다.

그래서 첫 번째 목표는 단순했습니다. 다음 질문서가 도착했을 때 더는 처음부터 다시 시작하고 싶지 않았습니다.

그럴듯하지만 틀린 답

가장 뻔한 해결책은 이전 질문서를 보관해 두었다가 그 답을 다시 쓰는 것이었습니다.

하지만 곧 그것만으로는 부족한 이유를 알게 됐습니다.

받은 문서 중에는 이미 답이 적혀 있는 것도 있었습니다. 그 답은 서식 예문이나 다른 공급업체의 답변, 또는 과거의 대화에서 가져온 것일 수 있었습니다. 그중 한 문서에는 ISO 9001 인증을 2017년에 받았다고 적혀 있었습니다. 바로 가져다 쓰기에 전혀 문제가 없어 보이는 문장이었습니다. 구체적이었고, 이미 잘 다듬어져 있었으며, 알맞은 질문 옆에 놓여 있었습니다.

정확한 연도는 2018년이었습니다.

한 해 차이에 불과했지만, 이 오류는 훨씬 더 큰 문제를 드러냈습니다. 예전 답변은 지식처럼 보입니다. 단정적이고 신뢰감을 주는 형태를 갖추고 있습니다. AI가 문서 안에서 이런 답변을 만나면 그대로 가져다 쓰기 쉽습니다. 하지만 그 답이 문서에 있다는 사실만으로는 누가 썼는지, 무엇을 근거로 했는지, 실제로 우리 제품을 설명하는지 알 수 없습니다.

그때 저는 복사해서 쓸 답변 모음을 만들어서는 안 된다는 것을 깨달았습니다. 우리가 실제로 답할 수 있다고 아는 내용을 기억하는 체계를 만들어야 했습니다.

고객에게 받은 질문서는 여전히 가치가 있었습니다. 고객이 무엇을 알고 싶어 하는지 보여 주었기 때문입니다. 그러나 그 문서만으로 답이 옳다는 것을 입증할 수는 없었습니다. 이 구분은 컨텍스트의 원칙이 되었습니다. 고객 문서는 질문을 보존하고, 답변은 우리가 확인한 정보와 연결된 채로 있어야 합니다.

양식이 아니라 사실을 보존하기

우리는 제안요청서에서 다음 요청에도 활용할 수 있는 내용을 추출하기 시작했습니다.

질문을 더는 고객별이나 번호별로만 분류하지 않았습니다. 백업, 사용자 인증, 암호화, 하도급업체, 호스팅, 서비스 연속성처럼 주제별로 서로 연결했습니다. 그러면 한 잠재 고객의 문서에서 '14.6.l'로 불린 질문과 다른

고객의 문서에서 'CL-072'로 불린 질문도 같은 주제를 다룬다면 함께 묶을 수 있었습니다.

컨텍스트는 각 주제에 관해 검증된 사실과 그 출처를 보관했습니다. 답변의 근거는 공식 문서, 확인된 설정, 테스트 결과 또는 담당 팀의 확인일 수 있었습니다. 검증한 날짜와 우리가 공개할 수 있는 내용의 범위도 함께 기록했습니다.

이 작업이 언제나 완전한 답을 만들어 내는 것은 아니었습니다. 대신 무엇이 비어 있는지도 드러냈습니다.

정보가 없으면 시스템은 그 사실을 밝혀야 했습니다. 담당 팀에 보낼 구체적인 질문을 작성하고, 그 답으로 질문서의 어느 항목을 채울 수 있는지도 표시했습니다. 그럴듯한 문장 뒤에 정보의 부재가 숨어 버리는 일은 이제 없었습니다. 정보의 공백은 누군가가 처리할 수 있는 명확한 일이 되었습니다.

마지막으로, 답을 검증하는 데 필요한 내용과 고객에게 보낼 수 있는 내용을 분리했습니다. 설정의 세부 사항은 내부 검토자에게 꼭 필요하지만 공개 문서에는 들어가면 안 될 수도 있습니다. 그래서 컨텍스트는 잠재 고객에게 전달할 문장을 준비하는 동시에, 검증에 필요한 증거와 유보 사항을 별도로 보존했습니다.

목표는 조금씩 달라졌습니다. 당장의 질문서를 끝내는 것만이 목적이 아니었습니다. 검증된 답변 하나하나가 다음 질문서에 필요한 일을 줄여야 했습니다.

네 번째 질문서

네 번째 제안요청서에는 첫 번째보다 훨씬 많은 115개의 질문이 있었습니다.

이번에는 흩어진 파일들을 처음부터 뒤지지 않아도 됐습니다. 컨텍스트는 반복해서 등장하는 주제를 이미 알고 있었고, 그에 맞는 사실을 어디서 찾아야 하는지도 알고 있었습니다. 새로 자료를 수집하지 않고도, 축적된 기

억과 제품 출처를 바탕으로 완전한 답변 96개를 곧바로 작성할 수 있었습니다.

시스템은 문서의 나머지 항목도 분류했습니다. 근거가 탄탄하지만 마지막 확인이 필요한 답변이 있었습니다. 여전히 불확실한 답변도 있었습니다. 가진 정보만으로는 답할 수 없는 질문도 몇 개 있었습니다.

무엇보다도, 부족한 정보의 성격이 달라졌습니다. 누가 무엇을 아는지 무작정 찾느라 몇 시간씩 쓰지 않게 됐습니다. 아직 열린 쟁점은 계약 기간 같은 영업상의 결정이나, 일부 기술 사항에 대한 확인이었습니다. 남은 일은 눈에 보였고, 담당자를 정할 수 있었으며, 범위도 제한적이었습니다.

이 결과를 통해 제안요청서에 '자동으로 답한다'는 말의 진짜 의미를 이해하게 됐습니다. AI에게 모든 칸을 자신 있게 채우라고 하는 것이 아닙니다. 바로 준비할 수 있는 것은 준비하고, 답변마다 신뢰 수준을 보여 주며, 여전히 사람의 판단이 필요한 부분을 명확히 가려내는 일입니다.

10분이 100퍼센트의 확실성을 뜻하지는 않는다

제가 실제로 사용해 보니, 제안요청서의 80~90퍼센트를 준비하는 데 세~여섯 시간 걸리던 시간이 약 10분으로 줄었습니다.

이 수치는 오해를 부를 수 있습니다. 답변의 80~90퍼센트가 즉시 확실해지고 회사가 책임질 수 있는 상태가 된다는 뜻은 아닙니다. 문서가 대부분 준비된다는 뜻입니다. 이미 아는 답변은 제자리에 들어가고, 출처를 다시 찾으며, 취약한 부분은 표시하고, 빠진 정보는 구체적인 문의 사항으로 바꿉니다.

사람의 검토는 여전히 필수입니다. 다만 이제는 검토할 곳이 정확해집니다. 예전 파일을 뒤지고 같은 사람들에게 또 물어보는 데 시간을 쓰는 대신, 검토자는 회사가 약속할 내용을 확인하고, 무엇을 공개할지 판단하며, 새롭거나 불확실한 답변에 집중할 수 있습니다.

따라서 가장 큰 이득은 몇 시간이 몇 분으로 줄어든 데에만 있지 않습니다. 일의 성격 자체가 달라진 데 있습니다.

첫 번째 질문서에는 조사가 필요했습니다. 그다음 질문서들은 그 조사로 배운 모든 것을 활용했습니다. 잘못된 연도 하나는 주의 원칙이 되었습니다. 담당 팀에서 받은 확인은 다시 쓸 수 있는 사실이 되었습니다. 답이 없는 질문은 깜빡하고 빠뜨린 항목이 아니라 이미 알고 있는 공백이 되었습니다.

잠재 고객은 여전히 자신이 요청한 문서를 받습니다.

하지만 회사는 그 문서를 준비하며 배운 것을 더는 잃지 않습니다.

기억할 점 – 답변의 가치는 근거에서 나온다

예전 답변은 지식처럼 보입니다. 단정적이고 잘 다듬어져 있지만, 어디에서 왔고 언제 작성됐는지는 말해 주지 않습니다. 이미 채워진 양식보다 검증된 사실과 그 출처를 보존하는 편이 낫습니다. 정보의 공백을 명시하는 편이 그럴듯한 문장보다 낫습니다. 그러면 그 공백은 누군가에게 건넬 질문이 됩니다.

3.2

아직 형태를 갖추지 못한 생각으로 글쓰기

제가 블로그에 올리는 글은 잘 준비된 빈 화면을 컴퓨터 앞에서 마주하며 시작되는 경우가 거의 없습니다.

어려움을 겪거나, 도구를 써 보거나, 누군가와 대화를 나누거나, 생각이 바뀐 뒤에 아이디어가 떠오릅니다. 걷는 중이나 대중교통 안에서, 또는 소파에 앉아 있을 때 찾아오기도 합니다. 글로 쓸 만한 생각이라는 느낌은 들지만, 정확히 어떤 관점에서 풀어야 할지, 어떤 순서로 이야기해야 할지는 아직 모릅니다.

그럴 때 저는 흔히 휴대전화를 듭니다.

단순히 음성 메모를 녹음하거나 나중에 옮겨 적을 내용을 음성으로 입력하는 데 쓰는 것은 아닙니다. **AI**와 대화를 시작합니다. 제가 생각하는 동안 AI는 듣고, 답하고, 다시 질문합니다. 이 대화는 나중에 컴퓨터에서 다시 열게 될 바로 그 작업 환경으로 들어갑니다. 따라서 책상에서 멀리 떨어져 시작하더라도, 나중에 옮겨서 다시 해석해야 하는 고립된 메모를 만들지 않아도 됩니다.

형성 중인 생각을 모두 꺼내 놓기

저는 글을 쓴다는 단순한 목표로 컨텍스트를 만듭니다. 그리고 머릿속에 있는 것을 전부 쏟아냅니다.

때로는 두 시간 동안 첫 번째 이야기를 하다가 두 번째로 넘어가고, 다시 첫 번째로 돌아오기도 합니다. 경험 하나를 이야기하다가 결론으로 건너뛰고, 세부 사항을 덧붙이고, 새로운 가능성을 열어 둔 채 앞의 이야기를 잠시 내버려 둡니다. 같은 생각이 다른 모습으로 거듭 떠올라 네 번씩 반복할 때도 있습니다. 처음에는 어떤 주장을 했다가 한 시간 뒤 정반대로 말할 수도 있습니다. 대화하는 사이에 제 생각이 달라졌기 때문입니다.

이런 무질서는 시작하기 전에 막아야 할 실수가 아닙니다. 아이디어가 막 태어나는 동안에는 대개 이런 모습이 자연스럽습니다.

AI와 대화하면 너무 일찍 구조를 강요하지 않고 계속 생각할 수 있습니다. AI의 답을 듣고 어떤 기억을 더 구체적으로 설명하게 됩니다. 질문을 받고 제가 한 단계를 건너뛰었다는 것을 알아차립니다. 다시 표현된 말을 들으며 당연하다고 여겼던 내용이 아직 분명하지 않다는 것을 깨닫습니다. 저는 고치고, 뉘앙스를 더하고, 다른 방향으로 다시 나아갑니다.

컨텍스트는 이 말을 세션 출처로 보존합니다. 잘 다듬어진 문장만 골라 남기지 않습니다. 생각이 어떻게 움직였는지 이해할 수 있도록 반복, 망설임, 모순도 함께 간직합니다.

그런 다음 AI는 같은 아이디어를 네 번 표현한 내용을 서로 다른 네 가지 아이디어로 다루지 않고 하나로 연결합니다. 서로 양립할 수 없는 두 주장을 표시하고, 한 시간 뒤에 나온 뉘앙스를 찾아내며, 한 원칙을 잘 보여 주는 사례를 그 원칙과 연결합니다. 무질서를 없애는 것이 아니라, 다룰 수 있는 상태로 바꿉니다.

이 첫 단계 덕분에 너무 이른 선택에서 벗어날 수 있습니다. 어떤 문장으로 글을 시작할지, 한 아이디어를 별도의 절로 만들지, 사건을 어떤 순서로 이야기할지 아직 결정하지 않아도 됩니다. 우선은 머릿속을 스쳐 지나가는 생각을 놓치지 않으면서, 그 안에 있는 것을 모두 꺼내 놓습니다.

그래도 컨텍스트는 이 자료가 왜 이곳에 있는지 알고 있습니다. 목적지 없는 대화를 받는 것이 아닙니다. 앞으로 글로 만들어야 할, 아직 정리되지 않은 재료를 받습니다.

AI를 생각을 숙성시키는 동반자로 삼기

첫 재료를 모두 내려놓고 나면, 무엇이 빠졌는지 AI에게 묻습니다.

AI는 너무 성급한 주장, 한 번도 설명하지 않은 전제, 독자가 제 경험을 이미 안다는 가정이 있어야만 이해되는 대목을 찾아냅니다. 사례를 요청합니다. 서로 다른 시점에 표현된 두 아이디어를 연결합니다. 그럴듯해 보이

는 방향을 제안할 때도 있지만, 제가 겪은 일과 맞지 않으면 저는 거절합니다.

저는 답하고, 구체화하고, 바로잡습니다. 이 대화가 아직 글을 만들어 내는 것은 아닙니다. 글의 재료를 더 충실하게 만듭니다.

AI의 역할은 중요하지만 그 자리는 분명합니다. AI는 제가 쏟아낸 모든 내용을 바탕으로 저보다 빠르게 그 안의 긴장을 찾습니다. 그러나 제가 실제로 무엇을 주장하려는지는, 제가 말로 표현하기 전까지 알지 못합니다. AI는 질문을 열고, 그 질문 때문에 글의 방향을 바꿀 가치가 있는지는 제가 결정합니다.

이 대화는 약점을 발견하는 데 걸리는 시간을 줄여 줍니다. 혼자였다면 이틀 뒤에야 사례가 빠졌거나 결론에 필요한 한 단계가 설명되지 않았음을 알아차렸을 것입니다. AI는 아이디어를 아직 손쉽게 고칠 수 있을 때 그 발견을 앞당깁니다.

경험만으로 부족할 때 출처 더하기

제 글은 흔히 제가 겪은 일에서 출발합니다. 하지만 외부 자료도 여전히 나름의 역할을 합니다.

- 참고 자료는 제가 관찰한 현상에 이름을 붙여 줍니다.
- 문서는 어떤 작동 방식을 확인해 줍니다.
- 다른 저자는 비슷한 입장을 옹호하거나, 제가 마주치지 못한 한계를 보여 줍니다.
- 추가 조사를 통해 제 경험이 하나의 특수한 사례에 불과하다는 사실이 드러나기도 합니다.

이 문서들은 문서 출처로서 컨텍스트에 들어옵니다. 처음에 제가 한 말과는 역할이 다릅니다. 세션 출처는 제가 주장하는 내용과 제 경험에서 이해한 것을 보존합니다. 문서 출처는 그 경험과 대조해 볼 수 있는 자료를 제공합니다.

그러면 글에서 제가 관찰한 것, 출처가 입증하는 것, 여전히 해석에 머무는 것을 구분할 수 있습니다. 이 구분은 개인적인 경험을 일반 법칙처럼 제시하려는 흔한 유혹에서 글을 지켜 줍니다.

출처를 재사용할 수 있는 아이디어로 바꾸기

이 단계에서 컨텍스트는 출처에서 곧장 글의 개요로 넘어가지 않습니다. 먼저 원자료를 원자 노트로 바꿉니다.

각 노트는 대화나 문서에서 나온 독립적인 아이디어 하나를 안정된 형태로 담습니다.

- 제가 주장하고 싶은 내용
- 그 주장을 구체적으로 보여 주는 사례
- 적용 범위를 제한하는 뉘앙스
- 글에서 답해야 할 반론
- 제 경험을 뒷받침하거나 반박하는 외부 사실

노트에는 출처가 남습니다. 그래서 대화 전체를 다시 읽지 않고도 아이디어를 이해할 수 있고, 표현을 확인하거나 원래 맥락을 찾아야 할 때는 출처로 돌아갈 수 있습니다.

이 변환은 글을 준비하는 방식을 바꿉니다.

- 대화는 제 생각의 움직임을 따라갔지만, 이제 노트는 같은 아이디어를 다루는 내용을 한데 모읍니다. 한 시간 간격으로 등장했거나 서로 다른 두 출처에서 나온 내용도 마찬가지입니다.
- 중심 노트 하나가 글의 관점을 드러낼 수 있습니다.
- 다른 노트들은 여기에 사례, 증거, 반론, 결과를 더합니다.

이 노트들은 아직 문단도, 미리 정해진 개요도 아닙니다. 결과물에 종속되지 않은 채로 남습니다. 같은 아이디어가 이 글뿐 아니라 발표나 다른 작업에도 쓰일 수 있습니다.

초안을 쓰기 전에 개요 거치기

재료가 충분해지면 저는 개요를 요청합니다.

이 단계 덕분에 풍부한 대화에서 곧장 대화 순서만 따라가는 글로 넘어가지 않게 됩니다. 대화는 되돌아가고, 바로잡고, 연상하며 진행됩니다. 하지만 글은 그 대화도, 그 과정도 모르는 독자를 이끌어야 합니다.

개요는 이 흐름을 정합니다. 무엇을 먼저 이해해야 하는지, 핵심 사례를 어디에 놓을지, 어떤 반론을 다룰지, 결론에서 실제로 어디까지 말할 수 있는지를 보여 줍니다.

AI는 가진 자료를 바탕으로 구조를 제안합니다. 저는 순서를 바꾸고, 범위를 좁히고, 부차적인 가능성을 지나치게 부각하는 부분은 거절합니다. 두세 번 의견을 주고받으면 대개 명확한 방향을 얻을 수 있습니다.

그다음에 초안을 씁니다.

이 단계에서 AI는 모호한 지시 하나만 받고 글을 만들어 내지 않습니다. 제가 들려준 경험, 질문에 대한 답, 여러 출처에서 추출한 아이디어와 주제의 묶음, 그리고 제가 승인한 개요를 가지고 작업합니다. 반복을 하나로 모으고, 사례를 알맞은 자리에 놓고, 이리저리 굽이치는 말을 읽을 수 있는 문장으로 바꿉니다.

저는 결과를 다시 읽으며 세 가지 어긋남을 구체적으로 찾습니다.

- 사라진 뉘앙스
- 지나치게 단정적으로 바뀐 대목
- 아이디어를 왜곡하는 표현

그 부분들을 바로잡은 뒤, 어떤 버전을 공개할지 결정합니다.

짧은 작업에서 컨텍스트가 하는 일

글 한 편은 하루나 이틀이면 쓸 수 있습니다. 몇 년에 걸쳐 이어지는 제품이나 수십 개 장으로 이루어진 책과 같은 기억이 필요해 보이지는 않습니다.

바로 이 가벼운 사례가 기간만이 유일한 쟁점은 아니라는 사실을 보여 줍니다.

컨텍스트는 처음 꺼낸 말, 추가한 출처, 제기된 질문, 선택한 개요, 완성된 글을 한데 보존합니다. 작업이 중단돼도 그 상태를 다시 찾을 수 있습니다. 어떤 문장이 의심받으면 그 기원으로 돌아갈 수 있습니다. 쓰는 도중 떠오른 아이디어가 이 글의 범위를 넘어선다면, 따로 안정된 형태로 남겨 미래의 작업에 활용할 수 있습니다.

이 자료를 모두 영원히 보존할 필요는 없습니다. 작은 컨텍스트는 목표를 달성하고 글을 만든 뒤 보관 상태로 전환할 수 있습니다. 그 가치는 작업이 형성되는 동안 이어짐을 지켜 주고, 나중에 글이 왜 이런 방향을 택했는지 이해해야 할 때 흔적을 남기는 데 있습니다.

글을 공개하면 또 다른 일이 시작될 수 있습니다. 요약문을 준비하고, 여러 언어에 맞게 글을 바꾸고, 온라인 게시에 필요한 요소를 만드는 일입니다. 이 작업들은 이미 컨텍스트가 알고 있는 규칙에 따라 같은 결과물을 확장합니다.

전체 흐름은 단순합니다. 경험이 말이 되고, 그 말이 질문과 출처를 만나며, 재료가 개요가 되고, 개요가 글이 됩니다.

AI는 각 변환을 빠르게 하고, 컨텍스트는 한 변환에서 다음 변환으로 이어지는 실을 놓치지 않습니다.

이러한 흐름은 긴 글쓰기에만 해당하지 않습니다. 아이디어 하나가 휴대전화 속에서 잊히는 음성 메모보다 더 나은 대접을 받을 가치가 생기는 순간부터 시작됩니다.

기억할 점 – 무질서도 재료다

막 태어나는 생각은 반복되고, 스스로 모순되며, 되돌아갑니다. 이는 시작 전에 고쳐야 할 결함이 아니라 자연스러운 모습입니다. 컨텍스트는 이 말을 세션 출처로 보존한 뒤, 다룰 수 있는 상태로 바꿉니다. 독자는 대화의 순서를 그대로 따라갈 수 없기에, 대화와 글 사이에는 개요가 들어갑니다.

3.3

혼란 속에서 제품 주제를 발전시키기

앞의 두 장에서는 한 번에 하나의 주제를 처음부터 끝까지 따라갔습니다. 이번 장에서는 여러 주제와 요청, 서로 다른 시간대의 일이 동시에 움직이는 프로덕트 매니저의 평범한 하루를 다룹니다.

저는 회사의 유일한 프로덕트 매니저입니다. 따라서 제품과 관련된 정보의 상당 부분이 결국 제 앞을 지나갑니다.

하루 중에 한 고객이 제게 중요하게 들리는 말을 합니다.

- 몇 마디에 불과합니다.
- 다른 사용자들도 겪는 문제를 드러내는지, 그 고객만의 어려움인지는 아직 알 수 없지만 놓치고 싶지는 않습니다.

조금 뒤에는 한 업종의 쟁점이 어떻게 변하고 있는지를 다룬 백서를 입수합니다.

- 이번에는 몇 마디가 아니라 수십 쪽짜리 PDF에 자료가 담겨 있습니다.
- 무엇이 유용할지, 이 문서가 현장에서 제가 관찰한 내용을 뒷받침할지도 아직 모릅니다.

그다음에는 개발자들이 현재 개발 중인 기능에 관해 이야기합니다.

- 질문에 답하거나, 동작을 구체화하거나, 몇 주 전에 세운 가정을 다시 살펴봐야 합니다.

그사이 CPO는 제품의 다른 영역과 관련된 두 가지 선택에 관해 함께 점검하기를 원합니다.

마지막으로 고객 지원팀과 대화하다가 기능 설명 문서가 빠져 있다는 사실이 드러납니다.

- 그 문서에 들어갈 내용 일부는 아직 제 머릿속에만 있습니다.

하루가 끝날 무렵이면 저는 서너 가지, 때로는 그보다 많은 주제를 오갔습니다. 몇 마디짜리 인용문, 문서 하나, 개발 중인 기능에 관한 질문, 판단 요청, 문서화 필요를 한꺼번에 받은 셈입니다.

유난히 나쁜 하루가 아닙니다. 평범한 하루입니다.

크기도 목적지도 서로 다른 입력

프로덕트 매니지먼트의 혼란은 단순히 정보가 많아서만 생기지 않습니다. 정보의 성격이 서로 다르기 때문에 생깁니다.

이 다섯 가지 입력은 각각 제품 생애의 서로 다른 시점을 바라봅니다.

- 고객의 한마디는 아직 다른 신호로 이어질지 알 수 없는 미약한 신호입니다.
- PDF는 한 업종을 전체적으로 조망합니다.
- 개발자들과의 대화는 지금 우리가 만드는 것을 다룹니다.
- 고객 지원팀의 요청은 이미 존재하며 설명해야 하는 것을 다룹니다.
- CPO의 질문은 가능한 여러 미래를 비교하게 합니다.

이 요소들을 하나의 문서에 모두 넣을 이유는 없습니다.

- 같은 주제를 다루지도 않고, 제품 생애의 같은 시점에 놓여 있지도 않습니다.

단순히 메모 목록 하나에 모아 두면 정보는 보존되지만, 이해하는 일은 실제로 필요해질 날까지 미뤄집니다.

여러 컨텍스트를 사용하면 여러 갈래의 흐름을 동시에 이어 갈 수 있습니다.

- 인용문은 의미가 아직 불확실하더라도 관련 있어 보이는 주제로 들어갑니다.
- PDF는 출처와 함께 보존되며, 분석을 거쳐 실제로 유용한 몇 가지 아이디어인 원자 노트를 추출할 수 있습니다.
- 개발자들과 나눈 대화는 현재 진행 중인 기능 작업에 합쳐집니다.
- 고객 지원팀이 알려 준 공백은 두 회의 사이에 흘려들은 말로 남지 않고, 만들어야 할 결과물이 됩니다.

들어온 모든 것이 곧바로 지식으로 바뀌는 것은 아닙니다.

- 고객의 말은 끝내 모호하게 남을 수도 있고,
- 백서는 별로 관련이 없을 수도 있으며,
- 직관은 다음 주에 반박될 수도 있습니다.

컨텍스트가 들어온 정보 자체를 더 지적인 것으로 만들어 주지는 않습니다. 다만 그 정보에 자리와 출처를 주고, 알맞은 때 다시 찾을 가능성을 줍니다.

컨텍스트는 혼란을 없앤다고 주장하지 않으면서, 혼란을 자기 안에 담아 둡니다.

정보가 들어오는 동안에도 결과물에 대한 요청은 계속된다

하루는 자료를 받고 분류하는 일로 끝나지 않습니다. 무언가를 만들어 내야 합니다.

예를 들어 고객 지원팀은 우리가 작업 중인 기능의 설명 문서를 요청합니다.

- 기능이 어떻게 작동하는지 설명하고, 중요한 경우를 구체적으로 밝히며, 고객을 지원하는 사람들이 올바르게 답할 수 있도록 정보를 제공해야 합니다.

같은 날 회사 블로그에 올릴 글을 마무리하려 할 수도 있습니다.

- 그 글이 반드시 같은 기능을 다루는 것은 아닙니다.
- 그래도 제품이 대응하는 업무, 사용자들이 되풀이해 겪는 어려움, 업무 방식의 변화처럼 같은 기능적 주제를 다룰 수는 있습니다.

두 문서는 같은 지식의 일부를 활용하지만, 같은 이야기를 하지는 않습니다.

- 고객 지원팀을 위한 문서는 제품이 실제로 어떻게 작동하는지 설명합니다.
  - 정확하고 실무적이어야 하며 사용자의 질문에 맞아야 합니다.
- 글은 경험이나 업무상의 문제, 또는 생각을 전개합니다.

  - 같은 아이디어 일부를 사용할 수 있지만, 다른 독자와 다른 목적에 맞게 구성합니다.

그 주제에 관해 제가 아는 내용을 두 번 다시 만들 필요는 없습니다.

- 컨텍스트는 아직 문서도 글도 아닌 중간 재료인 원자 노트와 주제 노트를 보존합니다.
- AI는 이 재료로 여러 구성을 준비할 수 있습니다. 하나는 제품 사용법에, 다른 하나는 더 넓은 기능적 쟁점에 초점을 맞춥니다.

물론 저는 두 결과물을 모두 다시 읽습니다.

- 고객 지원 문서에는 제품을 제대로 이해했는지가 걸려 있습니다.
- 글에는 제 관점이 걸려 있습니다.

하지만 같은 내용을 다시 찾아 처음부터 또 쓰는 대신, 내용과 방향이 옳은지 확인하는 데 시간을 씁니다.

이 흐름은 하루 안에서도 일어날 수 있습니다. 새로운 정보가 들어오는 동안, 이미 정리된 지식을 활용해 회사나 제게 필요한 문서를 만듭니다.

짧은 한마디가 제품 주제가 될 수 있다

아침에 들은 말이 반드시 곧바로 결정으로 이어지는 것은 아닙니다.

- 그 한마디만으로는 아무것도 입증하지 못할 수 있습니다.
- 몇 주 뒤 다른 고객이 비슷한 어려움을 이야기할지도 모릅니다.
- 인터뷰에서 새로운 뉘앙스가 더해질 수도 있습니다.
- 그날 입수한 백서가 관찰된 변화에 이름을 붙여 줄 수도 있습니다.
- 고객 지원 데이터가 우리가 생각한 것보다 문제가 더 자주 발생한다는 사실을 보여 줄 수도 있습니다.

이 요소들은 서로 다른 때, 서로 다른 형태로 들어옵니다.

- 컨텍스트를 사용하면 출처를 지우지 않으면서 서로 연결할 수 있습니다.
- 컨텍스트는
  - 새로운 의견이 이미 보존된 신호와 비슷하다는 점을 찾아내고,
  - 모순을 드러내며,

- 두 해석 가운데 어느 쪽이 맞는지 판단하려면 무엇이 필요한지 물을 수 있습니다.

고립된 한마디도 시간이 지나면 검토할 가치가 있는 주제가 될 수 있습니다. 그때가 오면 초기 신호, 그 신호를 비춰 주는 문서, 뒷받침하거나 반박하는 내용, 아직 열린 질문까지 그 역사의 일부가 이미 제 손에 있습니다.

컨텍스트는 이 재료가 무르익도록 돕습니다.
- 문제가 중요하다고 결정하지도, 저 대신 해결책을 고르지도 않습니다.

제품과 관련해 어떤 선택을 할지는 여전히 제 책임입니다.
- 다만 막연한 기억이나 흩어진 문서 모음이 아닌 더 나은 근거를 가지고 결정할 수 있습니다.

그다음 작업도 같은 시스템 안에서 이어집니다.
- 인터뷰를 추가하고,
- 사실에 비추어 가설을 검토하고,
- 여러 시나리오를 준비하고,
- CPO를 설득할 근거를 만들고,
- 명세를 작성하고,
- 그 밖의 일을 이어 갑니다.

하루의 혼란 속에서 포착한 작은 신호 하나가 실제 결정이 될 때까지 충분히 오래 살아남을 수 있습니다.

포기한 아이디어도 계속 무르익을 수 있다

모든 탐색이 실제 제품 출시로 이어지는 것은 아닙니다.

흥미로워 보이는 인공지능 활용 방안을 검토했지만, 당시 기술로는 제대로 구현할 수 없었던 적이 있습니다.
- 아이디어 자체가 반드시 나빴던 것은 아닙니다.
- 때가 맞지 않았습니다.

일반적인 방식이었다면 몇 가지 문서가 만들어진 뒤, 포기하기로 결정한 순간 어느 폴더 안으로 사라졌을 것입니다.

- 한두 해 뒤에는 그 파일들을 다시 찾고, 처음 세운 가정을 떠올리며, 왜 중단했는지 파악해야 했을 것입니다.

컨텍스트를 사용하면 최초의 신호, 상상했던 시나리오, 실험, 부딪힌 한계, 중단한 이유까지 탐색 과정과 함께 보관할 수 있습니다.

- 기술이 발전하면 과거의 부정적인 결과를 영원히 나쁜 아이디어로 단정하지 않고 그 작업을 다시 시작할 수 있습니다.

우리가 무엇을 하려 했는지뿐 아니라 왜 당시에는 되지 않았는지도 다시 찾을 수 있습니다. 그러면 장애물이 아직 존재하는지 확인하고, 여전히 유효한 부분은 유지한 채, 그 지점에서 생각을 이어 갈 수 있습니다.

제품 주제가 무르익는 과정이 언제나 끊김 없이 이어지는 것은 아닙니다.

- 한마디에서 시작해 몇 달 동안 풍부해지다가 멈추고, 제품이나 시장, 기술이 달라졌을 때 다시 시작될 수 있습니다.

하루의 속도를 늦추지 않고 질서를 끌어내기

제 하루는 여전히 조각나 있습니다.

- 고객들은 고객 지원팀의 요청에 맞춰 의견을 내주지 않습니다.
- 개발자들은 제가 마침 PDF를 다 읽었을 때를 골라 이야기하지 않습니다.
- 문서나 글을 쓰는 동안에도 준비해야 할 결정은 멈추지 않습니다.

컨텍스트가 이런 현실을 완벽하게 정돈된 작업 순서로 바꾸지는 않습니다.

대신 각각의 조각이 연속성을 유지하도록 합니다.

- 짧은 한마디와 긴 문서는 서로 다른 방식으로 처리하면서 받아들일 수 있습니다.
- 여러 주제가 뒤섞이지 않은 채 나란히 진행될 수 있습니다.
- 이미 다듬은 지식은 첫 번째 결과물 안에 갇히지 않고 여러 결과물에 쓰일 수 있습니다.

- 중단된 아이디어는 역사를 모두 잃지 않은 채 기다릴 수 있습니다.

제가 프로덕트 매니지먼트에서 컨텍스트를 사용하는 방식은 이렇습니다. 제 하루의 변화를 줄이기 위해서가 아니라, 그 변화 때문에 배운 것이 흩어지지 않게 하기 위해서입니다.

혼란은 계속 들어옵니다.

하지만 더는 손실이 되어 빠져나가지 않습니다.

이 세 가지 활용 사례에는 아직 이름 붙이지 않은 공통점이 있습니다. 모두 무언가가 바깥으로 나가면서 끝난다는 것입니다. 발송된 답변, 공개된 글, 내려진 결정이 그렇습니다.

다음 부에서는 결과물이 떠난 뒤 무엇이 남는지 살펴봅니다.

기억할 점 – 여러 흐름을 동시에 이어 가기

고객의 한마디와 50쪽짜리 PDF는 서로 다른 처리 방식과 목적지를 요구합니다. 컨텍스트는 각 조각에 자리와 출처를 주고, 알맞은 때 다시 찾을 가능성을 줍니다. 중단된 탐색도 멈춘 이유를 간직합니다. 기술이나 시장이 달라지면 다시 이어 갈 수 있습니다.

부 4

결과물이 남기고 간 것

앞의 세 가지 활용 사례는 각각 하나의 결과물을 전달하며 끝납니다. 이 부에서는 그 결과물이 떠난 뒤에 무엇이 남는지 살펴봅니다.

같은 질문을 6개월 간격으로 두 번 던지며 이야기를 이어 갑니다. 첫 번째에는 답이 스프레드시트의 셀 하나에 담깁니다. 두 번째에도 그 답은 남아 있지만 더 이상 충분하지 않습니다. 부족한 것은 그 답을 만든 추론, 그리고 그 추론을 뒷받침한 문서들 사이의 관계입니다.

4.1

답은 셀 하나에 담기지만 추론은 담기지 않는다

앞 부에서 다룬 제안요청서 중 하나로 돌아가 봅시다.

스프레드시트의 D47 셀에 다음 질문이 들어옵니다.

> 백업 데이터는 저장 상태에서 암호화됩니까?

최종 답변은 세 줄에 담깁니다. 그 세 줄을 쓰려면 여러 요소를 모아야 합니다.

- 문서에는 백업의 일반적인 작동 방식이 설명되어 있습니다.
- 설정 파일에는 실제로 활성화된 설정이 나타납니다.
- Ops 팀은 이 설정이 운영 환경에서 어떻게 적용되는지 확인해 줍니다.
- 법무팀은 잠재 고객에게 어떤 내용을 알릴 수 있는지 명확히 합니다.

저는 스프레드시트에서 문서로, 문서에서 설정으로 옮겨 간 뒤, Ops 팀과의 대화에서 법무적 판단으로 넘어갑니다. 마지막으로 D47 셀에 돌아와 답을 적습니다.

문서는 전송됩니다.

여러 단계를 거쳐 나온 세 줄

셀에는 작업의 결과가 담겨 있습니다. 그 결과를 만든 과정은 보여 주지 않습니다.

셀은 다음을 말해 주지 않습니다.

- 어떤 문서를 참고했는지,
- 어떤 설정을 확인했는지,
- 누가 설정의 적용을 확인했는지,
- 어떤 표현이 허용됐는지,
- 잠재 고객에게 보낸 답변에서 어떤 유보 사항을 제외했는지 말해 주지 않습니다.

그러나 이 요소들은 실제로 존재합니다. 이 요소들이 세 줄의 내용을 결정했고, 제가 답하는 동안에는 그 관계가 제 머릿속에 명확합니다.

문서를 보내고 나면 하나로 이어져 있던 이 전체가 사라집니다. 스프레드시트에는 답은 남지만, 그 답을 신뢰하게 한 추론은 남지 않습니다.

각 도구는 자신의 일을 했다

문제는 정리를 잘못한 데서 나오지 않습니다.

- 문서는 문서 공간에 잘 보관되어 있습니다.
- 설정은 제품 안에 남아 있습니다.
- 팀의 답은 해당 대화 스레드에 남아 있습니다.
- 법무적 판단은 그 자체의 문서에 담겨 있습니다.
- D47 셀은 잠재 고객의 문서 안에 제자리를 차지합니다.

각 도구는 자신에게 맡겨진 대상을 보존했습니다. 추론 전체를 보존하는 일은 어느 도구의 역할도 아니었습니다.

그 과정을 다시 구성하려면 무엇을, 어디서 찾아야 하는지, 그리고 각 요소가 어떤 역할을 했는지 알아야 합니다. 이 지식은 답을 준비한 사람이 지니고 있습니다.

결과물은 그것을 만든 과정을 평면화한다

결과물은 받는 사람에게 맞춘 결과를 보여 줘야 합니다.

- 잠재 고객은 명확한 답을 원하지, 그 답을 얻기 위해 조사한 전 과정의 기록을 원하지는 않습니다.

이런 단순화는 필요합니다. 그러나 결과물만이 작업의 유일한 흔적으로 남으면 비용이 발생합니다.

최종 결과는 그곳에 이르는 길보다 항상 짧습니다. 보낸 답도, 공개한 글도, 제품 결정도 그렇습니다. 결과만 남으면 다음 작업은 그 길의 일부를 다시 걸어야 합니다.

추론에 자리를 마련하기

컨텍스트는 스프레드시트, 문서, 설정, 팀과의 대화 중 어느 것도 대체하지 않습니다.

- 그것들을 잇는 작업에 공통의 자리를 마련합니다.

D47의 질문, 검토한 요소, 얻은 승인, 만든 답은 모두 같은 주제에 속합니다. 각자의 원래 형태와 위치를 유지하면서도 컨텍스트 안에서 서로 연결된 채 남을 수 있습니다.

셀은 계속 자신의 역할, 즉 잠재 고객에게 답하는 역할을 합니다. 컨텍스트는 다른 역할을 합니다. 그 답을 이해하고, 확인하고, 작업을 재개하는 데 필요한 것을 보존합니다.

6개월 뒤에는 이 차이가 결정적이 됩니다.

기억할 점 – 결과물은 추론을 보존하지 않는다

하나의 답이 세 줄에 담겨도 여러 출처와 승인, 단서에 의존할 수 있습니다. 각 도구는 자신의 대상을 보존하지만, 그 사이의 관계는 여전히 사람이 짊어집니다. 컨텍스트는 원래 도구를 대체하지 않고, 이 요소를 잇는 작업에 자리를 마련합니다.

4.2

6개월 뒤, 답만으로는 충분하지 않다

D47 셀을 작성한 지 6개월 뒤, 새로운 잠재 고객이 백업 암호화에 관해 같은 질문을 합니다.

- 10초 만에 예전 답을 찾습니다.
- 명확하고 잘 쓰였으며 바로 복사할 수 있습니다.

그러나 저는 그 답 앞에서 그 답을 쓰지 못하고 멈춰 있습니다.

세 가지 질문만으로도 다시 시작하는 일이 막힙니다.

- 이 답은 어디서 나왔는가? 문서인가, 직접 확인한 설정인가, 아니면 그날 Ops 팀이 설명한 내용인가?
- 법무팀은 이 표현 그대로를 승인했는가, 아니면 더 신중한 버전을 승인했는가?
- 오늘도 설정은 같은가?

답은 살아남았지만, 그 답을 신뢰할 수 있게 한 것은 사라졌습니다.

찾는 것과 재개하는 것은 다르다

예전 셀은 저에게 하나의 글을 되돌려 줍니다. 다음은 되돌려 주지 않습니다.

- 증거의 수준,
- 그 한계,
- 다시 사용하기 전에 필요한 확인입니다.

다른 기억이 없다면 앞 장의 조사를 거꾸로 되짚으며 전부 다시 해야 합니다. 이미 답이 있으므로 완전히 원점에서 출발하는 것은 아닙니다. 그럼에도 이미 한 일 중 어느 부분이 여전히 유효한지 모른 채 다시 시작합니다.

이것은 2018년 인증을 2017년으로 잘못 적은 오류를 우리 자신의 답 측면에서 바라본 것입니다. 문장은 명확하고 우리가 쓴 것이지만, 그 증거는 보이지 않게 됐습니다.

각 요소의 역할을 다시 찾기

컨텍스트 안에서 예전 답은 혼자 남지 않습니다. 그 답을 만든 추론에 필요한 관계도 함께 보존합니다.

```
문서 — 설명 ─┐
설정 — 확인 ─┼─► D47 답변
Ops 팀 — 보완 ─┤
법무팀 — 허용 ─┘
```

동사는 각 요소로 무엇을 할 수 있는지 바꿔 놓습니다.

- 문서는 일반적인 작동 방식을 설명하지만, 현재 활성화된 설정을 혼자서 증명하지는 못합니다.
- 설정은 특정 시점의 상태를 확인하지만, 무엇을 알릴 수 있는지 결정하지는 않습니다.
- Ops 팀은 실제 운영에서 얻은 보충 정보를 줍니다.
- 법무팀은 잠재 고객을 위한 표현을 허용합니다.

이제 저는 답 주변에 있는 네 개의 대상만 다시 찾는 것이 아닙니다. 각 대상이 어떤 기능을 했는지도 다시 찾습니다.

이런 정확성은 출처가 실제로 밝힌 것보다 더 많은 것을 증명하는 일을 막습니다. 또한 시간이 흐른 뒤 어떤 확인을 다시 해야 하는지 보여 줍니다.

바뀌었을 수 있는 것을 확인하기

컨텍스트는 예전 답을 영원한 진실로 바꾸지 않습니다. 검증을 준비합니다.

- 문서는 여전히 유효하지만 설정은 바뀌었을 수 있습니다.
- 설정은 같지만 허용된 표현은 다시 검토해야 할 수 있습니다.
- 제품이 발전하면서 Ops 팀이 더한 보충 정보가 더 이상 적용되지 않을 수 있습니다.

관계는 이런 차이를 눈에 보이게 합니다. 저는 전체 조사를 닥치는 대로 다시 하지 않고, 바뀌었을 수 있는 요소를 확인할 수 있습니다.

최종 답은 6개월 전과 같을 수 있습니다. 차이는 왜 그 답을 아직도 사용해도 되는지 알고 있다는 데 있습니다.

작업에 필요한 관계를 보존하기

모든 문서를 서로 연결하려는 것은 아닙니다.

- 두 파일에 같은 단어가 들어 있어도 같은 추론에 참여하는 것은 아닐 수 있습니다.

이해, 확인, 결정, 생산 방식을 바꾸는 관계라면 보존할 가치가 있습니다.

- D47의 경우에는 설명하기, 확인하기, 보완하기, 허용하기라는 네 관계면 충분합니다.

이 관계로 구체적인 질문에 답할 수 있습니다.

- 이 답은 무엇에 근거하는가?
- 오늘 어느 부분을 확인해야 하는가?
- 우리가 알릴 수 있는 내용을 누가 승인했는가?
- 어떤 변화가 이 답을 무효로 만드는가?

그러므로 컨텍스트는 제가 아는 모든 것의 완전한 지도를 만들려고 하지 않습니다. 특정한 작업에 필요한 연결망을 보존합니다.

시스템을 열어 보기

1권은 컨텍스트가 어떻게 태어났는지, AI와의 관계를 어떻게 바꾸는지, 여러 사용 방식을 어떻게 돕는지 보여 주었습니다. D47 사례는 아직 이해해야 할 것을 드러냅니다.

- 세션마다 반복하지 않고 작업의 임무를 어떻게 보존할 것인가?
- 출처, 아이디어, 결정, 결과물의 역할이 서로 구분되도록 어디에 놓을 것인가?
- 모든 자료를 대화에 넣지 않고 지금 필요한 것만 어떻게 불러올 것인가?

2권에서는 이 시스템을 열어 보겠습니다. 자료가 컨텍스트에 들어오는 순간부터 변환, 결과물 생성, 그리고 작업 재개에 이르기까지의 전 과정을 따라갑니다.

기억할 점 – 신뢰할 수 있게 하는 것을 다시 찾기

답을 다시 찾는 것만으로는 충분하지 않습니다. 그 답을 여전히 사용해도 되게 하는 것을 찾아야 합니다. 이름 붙은 관계는 출처, 승인, 단서의 정확한 역할을 보여 줍니다. 컨텍스트는 무엇이 답을 받쳐 주는지, 그리고 무엇이 바뀌었을 수 있는지 보여 줌으로써 검증을 준비합니다.

권 II

컨텍스트를 만들고 살아 있게 하기

5 컨텍스트 안을 들여다보기
6 생태계 안의 컨텍스트

부 5

컨텍스트 안을 들여다보기

지금까지는 컨텍스트가 할 수 있는 일을 중심으로 설명했습니다. 이 부에서는 그 안을 열어 봅니다.

먼저 전체 폴더 구조를 보여 준 뒤, 하나의 작업 세션을 처음부터 끝까지 따라갑니다. 열어서 현재 상태를 찾고, 자료를 넣고, 아이디어로 변환하고, 결과물을 만들고, 결정한 뒤, 다음 세션이 아무것도 다시 만들 필요 없이 시작할 수 있도록 저장합니다.

여섯 장을 관통하는 예시는 완전히 가상의 사례입니다. 일부 현장 기술자가 왜 불완전한 작업 보고서를 만드는지 이해한 뒤, 제품을 바꿀지 결정하려는 상황입니다.

5.1

전체 지도

2부에서 작업 보고서의 품질을 다루는 컨텍스트를 다음과 같은 상태로 남겨 두었습니다.

- 출처 세 개,
- 원자 노트 두 개,
- 주제 노트 하나,
- 다음 인터뷰를 위한 가이드 하나.

당시에는 이 컨텍스트를 통해 무엇을 이해할 수 있는지 살펴봤습니다. 이제 폴더 자체를 열어 보겠습니다.

이 책에서 사용하는 버전의 전체 폴더 구조는 다음과 같습니다.

```
contexte-qualite-comptes-rendus/
├── README.md                 자동 생성된 전체 탐색 안내
├── offline.md                세션 밖에서 쓰는 우편함
│
├── 00-frame/                 지속적인 방향을 정하는 것
│   ├── README.md             자동 생성된 이 영역의 탐색 안내
│   ├── mission.md            목표, 질문, 한계
│   ├── directives.md         임무에 고유한 규칙
│   └── loading.md            열 때 읽을 내용
│
├── 01-state/                 작업의 현재 상태
│   ├── README.md             자동 생성된 이 영역의 탐색 안내
│   ├── state.md              확립된 내용, 불확실성, 부채
│   ├── synthesis.md          지속적인 지식이 된 것
│   ├── todo.md               해야 할 일
│   ├── decisions.md          컨텍스트의 구조를 정하는 선택
│   ├── contradictions.md     아직 긴장 관계에 있는 것
│   ├── handoff.md            작업 재개 안내서
│   ├── checkpoints.md        상태를 압축해 기록한 연속 항목
│   └── health.md             마지막 점검 결과
│
├── 02-history/               날짜가 기록된 사실의 흔적
│   ├── README.md             자동 생성된 이 기록의 역할
│   └── YYYY-MM-DD.md         그날의 사건
│
├── 03-sources/               받아들이거나 표현한 것
│   ├── README.md             자동 생성된 출처 목록
│   ├── ingestion-log.md      읽은 내용과 거기서 나온 것
│   ├── documents/
│   │   ├── README.md         자동 생성된 문서 목록
│   │   ├── raw/              원본 파일
│   │   ├── fulltext/         활용할 수 있는 전체 텍스트
│   │   └── SRC00001-….md     신원, 출처, 상태
│   └── sessions/
│       ├── README.md         자동 생성된 세션 목록
│       └── SESSION-….md      작업 중에 제공한 말
│
├── 04-notes/                 출처에서 얻은 지식
│   ├── README.md             자동 생성된 노트 지도
│   ├── atomic/               노트 하나에 아이디어 하나 – ATO00001-….md
│   ├── thematic/             아이디어를 잇는 관점 – THM00001-….md
│   ├── glossary/             의미가 확립된 용어 – GLO00001-….md
│   └── decision-records/     판단 하나 – DRE00001-….md
│
└── 05-outputs/               특정 용도에 맞춰 만든 것
    ├── README.md             자동 생성된 결과물 목록
    ├── _draft/               작성 중인 결과물
    ├── questions-entretien.md   다음 단계를 위한 인터뷰 가이드
    ├── note-de-cadrage.md       결정을 위한 요약
    └── presentation-equipe.md   같은 지식을 다른 말로 풀어낸 자료
```

번호는 읽는 순서를 정할 뿐, 가치의 위계를 나타내지는 않습니다.

- 프레임과 상태는 작업의 방향을 잡습니다.

- 이력은 작업이 어떻게 진행됐는지 확인하게 해 줍니다.
- 출처는 재료를 보존하고, 노트는 그 재료를 지식으로 바꿉니다.
- 결과물은 그 지식을 받을 사람에게 맞는 형태로 만듭니다.

여섯 영역, 여섯 기능

`00-frame/`은 "나는 왜 여기 있으며, 어떤 규칙을 따라야 하는가?"라는 질문에 답합니다. 이 영역이 거의 바뀌지 않는 데에는 이유가 있습니다. 임무를 바꾸면 모든 출처와 작업, 결과를 판단하는 방식도 달라지기 때문입니다.

```
contexte-qualite-comptes-rendus/
├── 00-frame/                 지속적인 방향을 정하는 것
│   ├── README.md             자동 생성된 이 영역의 탐색 안내
│   ├── mission.md            목표, 질문, 한계
│   ├── directives.md         임무에 고유한 규칙
│   └── loading.md            열 때 읽을 내용
```

`01-state/`는 "나는 지금 어디까지 왔는가?"라는 질문에 답합니다. 이 영역은 작업에 따라 움직이며, 확립된 내용과 가설, 앞으로 할 일과 이미 내린 결정, 일시적인 어려움과 계속 드러내 놓아야 할 모순을 구분하게 해 줍니다.

```
contexte-qualite-comptes-rendus/
├── 01-state/                 작업의 현재 상태
│   ├── README.md             자동 생성된 이 영역의 탐색 안내
│   ├── state.md              확립된 내용, 불확실성, 부채
│   ├── synthesis.md          지속적인 지식이 된 것
│   ├── todo.md               해야 할 일
│   ├── decisions.md          컨텍스트의 구조를 정하는 선택
│   ├── contradictions.md     아직 긴장 관계에 있는 것
│   ├── handoff.md            작업 재개 안내서
│   ├── checkpoints.md        상태를 압축해 기록한 연속 항목
│   └── health.md             마지막 점검 결과
```

`02-history/`는 "실제로 무슨 일이 있었는가?"라는 질문에 답합니다. 이 영역은 세션을 보기 좋게 이야기하는 곳이 아닙니다. 추가된 출처, 포기한 방향, 내린 결정, 새로 만든 노트처럼 유용한 사실을 보존합니다. 평소에는 거의 들여다보지 않지만, 현재 상태가 앞뒤가 맞지 않는다고 느껴지는 날에는 결정적인 역할을 합니다.

```
contexte-qualite-comptes-rendus/
├── 02-history/              날짜가 기록된 사실의 흔적
│   ├── README.md            자동 생성된 이 기록의 역할
│   └── YYYY-MM-DD.md        그날의 사건
```

03-sources/는 "나는 어떤 재료를 받았는가?"라는 질문에 답합니다. 외부 문서는 고정된 상태로 들어오고, 제 말은 세션이 진행될 때마다 들어옵니다. 어느 쪽이든 출처는 컨텍스트가 그 내용을 어떻게 이해했는지와 분리된 채로 남습니다.

```
contexte-qualite-comptes-rendus/
├── 03-sources/              받아들이거나 표현한 것
│   ├── README.md            자동 생성된 출처 목록
│   ├── ingestion-log.md     읽은 내용과 거기서 나온 것
│   ├── documents/
│   │   ├── README.md        자동 생성된 문서 목록
│   │   ├── raw/             원본 파일
│   │   ├── fulltext/        활용할 수 있는 전체 텍스트
│   │   └── SRC00001-….md    신원, 출처, 상태
│   └── sessions/
│       ├── README.md        자동 생성된 세션 목록
│       └── SESSION-….md     작업 중에 제공한 말
```

SRC00001-….md 카드는 문서 자체도, 문서 요약도 담지 않습니다. 그 문서에 관해 제가 알아야 할 모든 것에 답합니다.

- 어디서 왔으며 왜 추가했는지,
- 원본과 전체 텍스트를 어디서 찾을 수 있는지,
- 텍스트 추출에 성공했는지,
- 어느 정도 신뢰할 수 있는지,
- 이 출처를 읽고 이미 노트를 만들었는지입니다.

따라서 재료 자체를 건드리지 않고도 출처의 상태를 눈에 보이게 합니다.

04-notes/는 "나는 무엇을 배웠는가?"라는 질문에 답합니다.

- 원자 노트는 아이디어 하나를 안정된 형태로 담습니다.
- 주제 노트는 여러 아이디어를 연결합니다.
- 용어집은 한 단어의 의미를 분명히 합니다.
- 결정 기록은 판단의 이유를 보존합니다.

```
contexte-qualite-comptes-rendus/
├── 04-notes/                 출처에서 얻은 지식
│   ├── README.md             자동 생성된 노트 지도
│   ├── atomic/               노트 하나에 아이디어 하나 — ATO00001-….md
│   ├── thematic/             아이디어를 잇는 관점 — THM00001-….md
│   ├── glossary/             의미가 확립된 용어 — GLO00001-….md
│   └── decision-records/     판단 하나 — DRE00001-….md
```

`05-outputs/`는 "이 지식으로 무엇을 할 것인가?"라는 질문에 답합니다. 초안과 결과물은 독자가 기대하는 형태로 이곳에 머뭅니다.

- 인터뷰 가이드,
- 글,
- 명세,
- 발표 자료,
- 결정 등 여러 형태가 가능합니다.

```
contexte-qualite-comptes-rendus/
└── 05-outputs/               특정 용도에 맞춰 만든 것
    ├── README.md             자동 생성된 결과물 목록
    ├── _draft/               작성 중인 결과물
    ├── questions-entretien.md   다음 단계를 위한 인터뷰 가이드
    ├── note-de-cadrage.md       결정을 위한 요약
    └── presentation-equipe.md   같은 지식을 다른 말로 풀어낸 자료
```

아이디어의 세계: 여러 건축물에 쓰이는 벽돌

`04-notes/`와 `05-outputs/`의 구분에는 컨텍스트의 가장 큰 장점 중 하나가 담겨 있습니다. 노트와 용어집은 이곳에서 *아이디어의 세계*를 이룹니다. 더는 원자료가 아니지만 아직 누군가에게 전달할 결과물도 아닌, 무한히 다시 쓸 수 있는 지식의 벽돌입니다.

```
03-sources/
     │
     ▼
04-notes/                          아이디어의 세계
├── ATO00001                       안정된 형태로 담긴 아이디어
├── ATO00002                       안정된 형태로 담긴 또 다른 아이디어
└── THM00001                       두 아이디어의 관계에 대한 관점
     │
     ├──> questions-entretien.md
     ├──> note-de-cadrage.md
     └──> presentation-equipe.md
```

제 사례에서는 입력의 불편함과 구조화된 데이터의 필요성이 어떤 형태로 제시될지 알기 전에 먼저 안정된 아이디어로 정리됐습니다.

- 인터뷰 가이드는 여기서 질문을 끌어냅니다.
- 검토 문서는 여기서 쟁점을 끌어냅니다.
- 발표 자료는 여기서 팀을 위한 이야기를 끌어냅니다.

이 세 결과물은 자신이 사용하는 지식을 소유하지 않습니다.

- 특정한 필요와 대상, 형식에 맞춰 그 일부를 선택해 다시 구성할 뿐입니다.
- 따라서 하나의 결과물이 그것을 구성한 노트를 소진하지는 않습니다. 그 벽돌은 다른 건축물에도 계속 쓸 수 있습니다.

이런 중간 세계가 없다면 새로운 결과물을 만들 때마다 출처를 다시 열고, 관계를 다시 찾고, 이미 얻은 같은 지식을 다시 표현해야 할 것입니다.

컨텍스트는 이미 만들어진 것만 보존하지 않습니다. 새롭게 만들어 낼 수 있는 지속적인 역량을 구축합니다.

모든 파일이 같은 방식으로 변하지는 않는다

각 요소가 어떻게 변하는지 알아야 비로소 이 폴더 구조를 제대로 읽을 수 있습니다.

일부 파일은 지금 가능한 최선의 상태를 보여 주기 위해 다시 작성됩니다.

- `state.md`,
- `handoff.md`,

- `05-outputs/_draft/`의 초안입니다.

다른 파일들은 과거를 보존하기 위해 내용이 추가되기만 합니다.
- 세션 출처,
- 날짜별 이력,
- 종합 기록,
- 체크포인트의 연속 기록입니다.

받은 문서는 그대로 고정되며, 다른 노트로 대체된 노트도 계속 열어볼 수 있습니다. 마지막으로 `README.md` 파일은 실제로 존재하는 내용에서 자동으로 생성됩니다. 이를 직접 고치면 현실을 설명하는 두 번째 버전이 생기고, 결국 실제 상태와 어긋나게 됩니다.

이런 다양성 때문에 하나의 거대한 파일로는 더 이상 충분하지 않았습니다. 하나의 텍스트에 다음 역할을 모두 요구해야 했기 때문입니다.
- 현재를 보여 주고,
- 과거를 보존하고,
- 원자료를 받아들이고,
- 아이디어를 안정된 형태로 만들고,
- 완성된 결과를 담는 역할입니다.

그대로 베끼는 조리법이 아니라 골격

이 골격은 컨텍스트 옆에 놓인 설명서가 아니라, 제가 실제로 복사해 쓰는 폴더입니다.
- 각 파일에는 담아야 할 절과 몇 가지 안내가 미리 들어 있으므로, 몇 달 간격으로 만든 두 컨텍스트도 같은 이정표를 공유합니다.

그렇다고 구조를 바꿀 수 없는 것은 아닙니다. 컨텍스트에는 다음과 같은 것을 추가할 수 있습니다.
- 특별한 지침,
- 특정 업무에 고유한 결과물,
- 전에 없던 유형의 출처입니다.

공통 골격이 지키는 것은 무엇보다 제 작업에 필요한 세 가지 경계입니다.
- 안정된 것과 현재 상태,
- 원자료와 해석된 내용,
- 지식과 지식을 표현한 형태 사이의 경계입니다.

컨텍스트에 말 걸기

지금까지 폴더와 파일을 보여 드렸습니다. 그렇다면 제가 이 파일을 하나씩 열고, 어디에 무엇을 쓸지 알고, 모든 정보를 직접 옮겨야 한다고 생각할 수 있습니다. 하지만 저는 그렇게 일하지 않습니다.

AI에게 말합니다.

> 작업 보고서의 품질을 다루는 컨텍스트를 열어 줘.

> 이 문서를 출처로 추가해 줘.

> 이 출처를 노트로 변환해 줘.

> 작업을 저장해 줘.

각 요청 뒤에는 제가 한 번 작성해 두고 AI가 매 세션마다 다시 따를 수 있는 절차가 있습니다.
- Claude Code와 Codex에서는 이런 재사용 가능한 절차를 스킬이라고 부릅니다.

예를 들어 열기 스킬은 다음 내용을 정합니다.
- 시작하기 전에 무엇을 파악해야 하는지,
- 어떤 파일을 반드시 읽어야 하는지,
- 작업에 필요해질 때까지 무엇을 디스크에 그대로 두어야 하는지,
- 문제를 어떻게 알려야 하는지,
- 제가 곧바로 행동할 수 있도록 어떤 정보를 돌려줘야 하는지입니다.

매번 이 단계를 다시 설명할 필요는 없습니다.

- 그저 컨텍스트를 열어 달라고 요청하면 됩니다.

대화에는 지금 이 순간의 제 의도가 담깁니다.

- 스킬에는 되풀이되는 행동을 신뢰할 수 있게 실행하는 방법이 담깁니다.

따라서 제 시스템에는 몇 가지 안정된 동작이 있습니다.

- 컨텍스트 만들기,
- 열어서 진행 중인 작업 되찾기,
- 출처를 추가하거나 세션 보존하기,
- 출처를 노트로 변환하기,
- 작업 상태 저장하기,
- 일관성 점검하기,
- 목적을 달성하면 보관하기입니다.

AI와 나누는 모든 대화에 스킬을 사용하는 것은 아닙니다. 저는 여전히 일상 언어로 생각하고, 질문하고, 바로잡고, 판단합니다.

스킬은 언제나 같은 단계를 지켜야 하는 작업에만 틀을 제공합니다.

- 예를 들어 출처를 어디서 얻었는지 남기지 않은 채 변환하거나, 작업 재개를 준비하지 않고 세션을 끝내는 일을 막습니다.

절차는 안정적으로 유지되지만, 판단까지 자동화되는 것은 아닙니다. 두 모델이 같은 노트를 서로 다르게 표현할 수 있습니다. 그래도 같은 요소를 살펴보고, 같은 경계를 지키며, 같은 점검을 수행해야 합니다.

따라서 누군가와 스킬을 공유한다고 해서 그 사람에게 제 컨텍스트를 주는 것은 아닙니다. 자신만의 컨텍스트를 저와 같은 방식으로 만들고 계속 운영할 방법을 주는 것입니다.

스킬의 작동 원리는 여기서 더 설명하지 않겠습니다.

스킬은 컨텍스트가 발명한 것도, 컨텍스트에만 있는 것도 아닙니다. Claude Code와 Codex는 재사용 가능한 절차를 설명할 수 있도록 스킬을 제공하며, 만드는 법과 쓰는 법은 각 제품의 문서가 이 책보다 더 잘 설명합니다.

컨텍스트에 고유한 것은 어떤 능력을 부여할지 선택하는 일입니다. 만들기, 열기, 자료 넣기, 변환하기, 저장하기, 점검하기, 보관하기가 이에 해당합니다. 이 능력들이 이어지는 순서와 각 능력이 지켜야 할 경계도 컨텍스트의 몫입니다.

파일은 컨텍스트가 아는 것을 보존합니다. 스킬은 컨텍스트를 변화시키는 동작을 조직합니다.

지도가 펼쳐졌습니다. 이제 실제 세션처럼 컨텍스트를 여는 데서부터 살펴볼 수 있습니다.

기억할 점 – 컨텍스트의 골격

새로운 컨텍스트에는 이미 가구가 갖춰져 있습니다. 서로 다른 질문에 답하는 여섯 영역이 있습니다. 번호는 읽는 순서를 정할 뿐, 가치의 위계를 나타내지는 않습니다. 폴더 구조를 읽을 수 있게 하는 것은 모양이 아니라 각 파일이 어떻게 변하는지 아는 것입니다. Claude Code와 Codex는 스킬이라는 작동 방식을 제공하고, 컨텍스트는 반복되는 동작과 그 순서, 한계를 정의합니다.

5.2

열기: 임무와 현재 상태 되찾기

기술자가 설비 작업 뒤에 확인하고 수행한 내용을 기록하는 문서, 즉 작업 보고서를 마지막으로 다룬 지 2주가 지났습니다.

- 인터뷰와 노트는 여전히 제자리에 있습니다.

하지만 아무렇게나 다시 열어 보면 이미 알고 있던 내용을 재구성하느라 30분을 쓰게 되고, 그마저도 정확하지 않을 것입니다.

- 그래서 열기는 훨씬 작은 핵심에서 시작합니다. 프레임, 상태, 작업 재개 정보입니다.

제가 이 모든 것을 일일이 요청할 필요는 없습니다. 열기 스킬을 호출하면 정해진 절차가 실행됩니다.

먼저 내가 왜 여기 있는지 되찾는다

AI가 컨텍스트 열기 스킬을 통해 가장 먼저 읽는 파일은 `00-frame/mission.md` 입니다. 그 템플릿은 다섯 가지를 묻습니다.

```
# 임무

## 목표
이 `컨텍스트`를 통해 무엇을 할 수 있어야 하는가?

## 중심 질문
분류와 탐색의 방향을 잡는 질문은 무엇인가?

## 주제에 포함되는 것
어떤 자료를 받아들일 수 있는가?

## 주제에서 제외되는 것
흥미로워 보여도 무엇을 받아들이지 않을 것인가?

## 예상 결과물
이미 알고 있다면, 무엇을 만들려 하는가?
```

제 파일의 앞부분은 다음과 같이 채워져 있습니다.

```
## 목표
작업 보고서의 품질을 떨어뜨리는 요인을 이해한다.

## 중심 질문
입력하기 싫어질 정도로 번거롭게 만들지 않으면서 활용할 수 있는 정보를 어떻게 수집할 것인가?

## 주제에서 제외되는 것
- 여러 출처를 비교하기 전에 인터페이스를 선택하는 것.
- 인터뷰 하나만으로 일반화하는 것.

## 예상 결과물
제품 결정을 준비하고 다음 인터뷰를 설계한다.
```

"작업 보고서 개선"이라고만 적은 임무는 방향을 잡는 데 별 도움이 되지 않습니다. 반면 중심 질문은 어떤 문서가 다른 문서보다 더 관련 있는지 이미 가려 줍니다. 주제에서 제외되는 것을 정하면, 매력적인 아이디어 하나가 저도 모르게 탐색을 해결책 개발 프로젝트로 바꾸는 일을 막을 수 있습니다.

- 결과물은 아직 정해지지 않아도 됩니다.
- 저는 성급하게 결과물을 꾸며 내는 대신 이 불확실성을 적습니다. 그러면 컨텍스트는 아직 탐색 중이라는 사실을 압니다.

전에 정해 둔 규칙이 나를 기다린다

비슷한 두 임무에도 서로 다른 주의 사항이 필요할 수 있습니다. `directives.md`는 다음과 같은 규칙을 작업과 함께 보존합니다.

- 기밀 유지,
- 증거 수준,
- 용어,
- 검증 또는 요구되는 형식,
- 그 밖의 규칙입니다.

템플릿은 의도적으로 단순합니다. 규칙 하나, 날짜 하나, 명령형 문장 하나로 이루어집니다.

```
# 지침

- (2026-09-11) 인터뷰를 추가하기 전에 익명화한다. 인터뷰 참여자의 이름은 직무로 바꾸되, 회사
이름과
  국가는 유지한다.
- (2026-09-11) 출처가 보여 주는 내용과 우리가 추론한 내용을 구분한다.
- (2026-09-11) 실제 사용 방식을 문서화하기 전에는 어떤 해결책도 제안하지 않는다.
```

지침은 앞으로의 행동을 이끌 수 있어야만 가치가 있습니다.

- "주의할 것"이라고 쓰면 해석하는 일을 매번 다시 해야 합니다.
- "관찰과 추론을 구분할 것"이라고 쓰면 노트나 결과물을 작성할 때 확인할 수 있는 정확한 기준이 생깁니다.

그러나 지침은 파일을 열 때 다시 발견하는 안내문에 그치지 않습니다. 행동이 적용 범위에 들어올 때마다 컨텍스트가 따라야 하는 지속적인 규칙입니다.

예를 들어 새로운 인터뷰 녹취록을 추가한다고 합시다. 인터뷰 대상자의 이름이 들어 있다는 사실을 다시 언급하지 않고, 전용 스킬을 통해 컨텍스트에 통합해 달라고 요청하기만 합니다. 그런데도 추가 과정에서 익명화 지침이 적용됩니다. 스킬은 녹취록을 출처로 저장하기 전에 인터뷰 대상자의 이름을 직무로 바꾸고, 회사 이름과 국가는 그대로 둡니다.

규칙이 필요해지는 바로 그 순간에 제가 그 규칙을 떠올릴 필요는 없습니다. 컨텍스트가 대신 기억합니다.

따라서 열기가 끝난 뒤에도 지침은 계속 작용합니다. 세션 내내 적용되며 컨텍스트의 여러 동작을 따라다닙니다.

- 출처를 추가할 때 익명화 규칙이 인터뷰 대상자의 신원을 보호합니다.
- 노트를 만들 때 증거 수준에 관한 규칙이 출처가 보여 주는 것과 우리가 추론한 것을 구분합니다.
- 결과물을 준비할 때 마지막 지침이 해결책을 너무 일찍 확립된 사실처럼 제시하지 못하게 합니다.

작업을 실행하는 스킬은 먼저 관련 지침을 식별한 뒤, 결과가 그 지침을 따르는지 확인해야 합니다.

```
요청
  ↓
적용할 지침
  ↓
스킬의 작업
  ↓
결과 점검
```

이처럼 지침은 작업을 둘러쌉니다. 실행 전에는 작업을 제한하고, 실행 뒤에는 결과를 점검하는 기준이 됩니다. 지침이 스스로 작동하거나 모든 것을 아는 것은 아닙니다. 알맞은 때 지침이 실제로 작용하게 만드는 것은 스킬입니다.

임무는 어디로 갈지 알려 줍니다. 지침은 컨텍스트의 모든 동작을 따라다니며, 이미 정해 둔 주의 사항을 단계마다 제가 다시 떠올리지 않아도 되게 합니다.

모든 것을 다시 열 필요는 없다

제 폴더에는 수십 개의 출처와 노트가 있습니다. 열 때마다 전부 불러오면 방대한 양에 우선순위가 묻힐 것입니다. 그래서 `loading.md`는 안정적인 핵심만 선택합니다.

```
## 항상

- `00-frame/mission.md`
- `00-frame/directives.md`
- `01-state/state.md`
- `01-state/todo.md`
- `01-state/handoff.md`

## 필요할 때

- 확인에 필요한 출처
- 작업 중인 주제와 관련된 노트
- 현재 작성 중인 초안
```

모든 것에 접근할 수는 있지만, 모든 것을 AI의 컨텍스트에 불러오지는 않습니다. `loading.md`는 임무를 이해하고, 요청을 해석하며, 이미 정해진 규칙을 잃지 않고 행동하는 데 꼭 필요한 요소를 지정합니다.

- 나머지는 디스크에 남아 있다가 필요할 때만 읽습니다.
- 오래된 출처 하나가 어떤 주장을 확인하는 데 꼭 필요할 수 있지만, 그렇다고 모든 세션에서 활성 컨텍스트의 공간을 차지할 필요는 없습니다.

`loading.md`의 내용은 컨텍스트마다 다릅니다.

- 책이라면 열 때마다 목차를 불러오고, 제품 탐색이라면 중심이 되는 주제 노트를 불러올 수 있습니다.
- 원칙은 같습니다.
  - 행동할 수 있을 만큼 충분한 정보로 시작하고,
  - 작업상 필요할 때 나머지를 찾아갑니다.

실제로 어디까지 와 있었는가

임무는 안정적으로 유지되지만 `01-state/state.md`는 변하도록 만들어졌습니다. 세 부분으로 이루어집니다.

```
# 상태

## 알고 있는 것
- 첫 번째 인터뷰에서는 나중에 입력할수록 세부 정보가 사라지는 것으로 나타났다.
- 두 번째 인터뷰에서는 일부 필수 입력란을 우회하는 것으로 나타났다.

## 아직 불확실한 것
- 입력 뒤 실제로 어떤 항목이 다시 사용되는지는 알지 못한다.

## 부채
- 백서를 추가했지만 신뢰 수준은 아직 평가해야 한다.
```

상태 파일은 전체 역사를 이야기하지도, 모든 노트를 요약하지도 않습니다. 작업 자체에 관해 현재 제가 가진 최선의 이해를 보여 줍니다.

`state.md`는 AI와 사람 모두를 위한 파일입니다.

AI에게는 지속되는 작업 기억입니다.

- 열 때마다 불러오므로, 활용할 수 있는 확립된 내용과 그대로 보존해야 할 불확실성, 잊어서는 안 될 부채를 알려 줍니다.
- 이 파일이 없으면 AI는 새 세션을 시작할 때마다 출처와 노트에서 이 상태를 다시 구성해야 합니다.

저에게는 대시보드 역할을 합니다.

- 컨텍스트가 무엇을 확립된 사실로 보는지, 무엇을 여전히 불확실하게 여기는지, 어떤 작업이 미완성인지 몇 줄만으로 확인할 수 있습니다.
- 그 상태가 제 이해와 다르면 바로잡을 수 있습니다.

따라서 수십 줄만 있어도 AI는 확립된 내용과 가설을 혼동하지 않고 작업을 재개할 수 있고, 저는 AI가 의지하는 이해가 맞는지 확인할 수 있습니다.

todo.md는 여기서 해야 할 일을 추출합니다.

```
## 진행 중
- T-001 – 필수 입력란의 실제 사용 방식을 알아보기 위한 인터뷰 준비.

## 할 일
- T-002 – 감사에 어떤 데이터가 쓰이는지 확인.

## 완료
- T-003 – 첫 두 인터뷰 비교. – 2026-09-10 완료
```

제가 todo.md를 직접 수정하려고 열 필요는 없습니다. 그저 이렇게 말할 수 있습니다.

> 감사에 어떤 데이터가 쓰이는지 확인하는 작업을 추가해 줘.

컨텍스트는 한 줄을 만들고 식별자를 부여한 뒤 알맞은 절에 배치합니다.

작업 도중 할 일이 생길 수도 있습니다.

- 출처를 분석하다가 확인해야 할 사항이 드러나면, AI는 대화 속에서 사라지게 두지 않고 할 일로 추가할 것을 제안합니다.

- 모든 아이디어를 말없이 새로운 의무로 바꾸지는 않습니다. 할 일이 우선순위를 바꾸거나 추가 작업을 요구한다면 판단의 책임은 여전히 제게 있습니다.

일을 마치면 다음과 같이 요청할 수 있습니다.

> *T-002 작업을 완료 처리해 줘. (작업 내용을 대략 설명해도 AI가 해당 항목을 찾아 줍니다.)*

컨텍스트는 해당 줄을 삭제하지 않습니다. 완료로 옮기고 완료 날짜를 추가하며, 필요하다면 완료됐다고 판단할 수 있는 결과나 근거도 표시합니다.

- 완료된 작업도 사라지지 않습니다.
- 그 기록 덕분에 같은 일을 되풀이하지 않고, 어떤 방향이 더 이상 활성 상태가 아닌 이유도 이해할 수 있습니다.

따라서 `todo.md`는 한 번 작성하고 마는 목록이 아닙니다. 컨텍스트가 처리할 작업의 대기열입니다. AI는 남은 일을 파악하기 위해 이를 확인하고, 작업이 진행되면 갱신하며, 같은 일을 다시 하지 않도록 완료된 작업도 보존합니다.

어디서부터 다시 시작할 것인가

`handoff.md`는 핵심 파일 가운데 마지막입니다. 저장할 때마다 다시 작성되며, 아주 구체적인 상황에 답합니다. 사람이나 AI가 이전 대화를 전혀 모른 채 지금 도착했다고 합시다. 그 사람이 잘못된 방향으로 가지 않고 다시 시작하려면 무엇을 알아야 할까요?

```
# 작업 재개

## 현재 상태
세 출처에서 입력 편의성과 데이터 활용 사이의 긴장이 드러났다.

## 계속하기 전에 알아야 할 것
두 인터뷰는 개별적인 상황을 설명하는 반면, 백서는 일반적인 관점을 주장한다.
이 자료에서는 긴장이 드러나지만, 그 규모를 측정하거나 모든 조직에 존재한다고 결론 내릴 수는 아직 없다.

## 다음 행동
실제로 사용되는 입력란을 파악할 질문을 준비한다.

## 열린 채로 남은 것
감사에 필요한 구조화 수준은 아직 알 수 없다.
```

다음 행동은 하나뿐이며 곧바로 실행할 수 있어야 합니다. 열 가지 가능성을 나열한다면, 바로 그 우선순위 결정을 덜어 주기 위해 존재하는 파일이 다시 저에게 일을 떠넘기는 셈입니다.

따라서 열기 명령은 컨텍스트가 잠든 동안 들어온 내용을 받아들이고, 이 핵심 파일들을 읽은 뒤, 다음 행동을 알려 줍니다.

- 세부 사항보다 먼저 목적을 되찾으며, 제가 2주 동안 자리를 비운 대가는 1분만 읽으면 되는 정도로 줄어듭니다.
- 이제 세션은 방향을 잃지 않은 채 새로운 자료를 받아들일 수 있습니다.

기억할 점 – 열기의 핵심

컨텍스트를 다시 여는 것은 그 내용을 모두 다시 읽는 일이 아니라, 임무와 지침, 상태, 작업 재개 정보로 이루어진 작은 핵심을 불러오는 일입니다. 임무는 어디로 갈지 말하고, 지침은 그 여정의 조건을 정합니다. 유용한 작업 재개 정보는 곧바로 실행할 수 있는 단 하나의 행동을 제시합니다.

5.3

자료 넣기: 재료를 받아들이기

세션이 시작되자 세 가지가 동시에 들어옵니다.

- 운영 책임자가 인터뷰 보고서를 보내고,
- 대화 중에 아이디어 하나가 떠오르며,
- 전날 남겨 둔 메모 하나가 수신함(메모 앱, 이메일 등)에서 기다립니다.

이 세 가지 재료는 서로 다른 방식으로 컨텍스트에 들어오며, 각각에 맞는 스킬이 있습니다.

문서는 완성된 객체로 들어온다

외부 문서는 원본 파일, 활용 가능한 텍스트, 상태를 추적할 수 있는 카드와 함께 보존됩니다.

```
03-sources/documents/
├── SRC00001-entretien-exploitation.md
├── raw/
│   └── SRC00001-entretien-exploitation.pdf
└── fulltext/
    └── SRC00001-entretien-exploitation.md
```

`SRC00001-entretien-exploitation.md` 카드는 다음과 같은 형태가 될 수 있습니다.

```
---
added: 2026-09-11
kind: pdf
intention: entretien
raw: raw/SRC00001-entretien-exploitation.pdf
fulltext: fulltext/SRC00001-entretien-exploitation.md
extraction: ok
reliability: medium
---

# 운영 책임자 인터뷰

## 이곳에 있는 이유

현장 기술자가 활용 가능한 작업 보고서를 작성하지 못하게 하는 요인을 이해한다.

## 해석 – 분석 전

운영 책임자는 현장에서 빠르게 입력해야 한다는 요구와 분석에 필요한 구조화된 데이터 사이에 충돌이 있다고 설명한다.

## 다루는 내용

- 모바일 입력
- 필수 입력란
- 데이터 재사용

## 여기서 나온 것

- 분석 뒤 작성
```


이 카드가 반드시 문서를 요약하는 것은 아닙니다. 우선 관리에 필요한 질문에 답합니다.

- 어디서 왔는가,
- 왜 추가했는가,
- 내용을 읽을 수 있는가,
- 어느 정도 신뢰할 것인가입니다.

`해석 – 분석 전` 절은 최초의 해석을 격리합니다. 가져오는 순간 생긴 인상이 곧바로 확립된 지식으로 취급되는 일을 막습니다.

출처가 PDF나 웹 페이지처럼 곧바로 활용하기 어려운 형식으로 들어오면, 텍스트 추출을 통해 사람과 AI가 읽을 수 있는 버전을 만듭니다.

- 이 작업은 아직 어떤 지식도 만들어 내지 않습니다. 원본 문서를 바꾸지 않고 재료에 접근할 수 있게 할 뿐입니다.

변환은 그다음에 이루어집니다.

- 추출한 텍스트를 분석해 원자 노트, 주제 노트, 용어집 항목, 결정 기록을 찾아냅니다.
- 이 노트들은 *아이디어의 세계*를 채웁니다. 그곳에서 출처의 재료는 재사용할 수 있는 지식의 벽돌로 바뀝니다.

신뢰 수준은 추가된 출처의 좋고 나쁨을 평가하지 않습니다.

- 고립된 증언, 검증된 데이터, 계약 문서, 확인해야 할 가설처럼 그 재료를 어떻게 사용해야 하는지 알려 줄 뿐입니다.

유용한 재료가 모두 문서 형태로 이미 존재하는 것은 아닙니다. 일부는 제 경험과 기억, 관찰, 그리고 작업하면서 제가 만들어 내는 구분 속에 있습니다.

AI와 대화하는 세션 중에 제가 이를 말로 표현하면, 그 내용도 컨텍스트에 들어갑니다.

- 자동으로 확립된 사실이 되지는 않습니다.
- 출처를 유지하며 나중에 다른 출처와 대조할 수 있습니다.

따라서 재료를 추가한다는 것은 문서를 통합하는 일인 동시에, 컨텍스트의 소유자가 대화에 보태는 내용을 보존하는 일이기도 합니다.

사람의 지식은 세션을 통해 들어온다

세션은 한 번에 완성되어 전달되는 문서가 아닙니다. 대화가 이어지면서 만들어집니다. `03-sources/sessions/SESSION-2026-09-12.md`에 기록되며, 다음과 같은 형태가 될 수 있습니다.

```
# 2026년 9월 12일 세션

## 09:41 – AI

필수 입력란이 실제로 품질을 높이는가, 아니면 현장 기술자에게 부담을 떠넘길 뿐인가?

## 09:42 – 저자

인터뷰를 보면 필수 입력란은 무엇인가가 입력됐다는 사실만 보장하는 경우가 많다.
정확성이나 유용성까지 보장하지는 않는다.
현장 기술자가 어떤 입력란이 나중에 어떻게 쓰이는지 이해하지 못하면, 다음 단계로 넘어가기 위해 최대한 빨리
  채운다.
따라서 제약을 두면 겉으로 보이는 완성도는 높아지지만, 실제로 활용할 수 있는 정보를 개선하지 못한 채 부담만 옮길
  수 있다.
...
```

AI가 말한 모든 내용을 보존할 필요는 없습니다. 하지만 작업을 바꾸는 질문, 반론, 재표현은 보존할 가치가 있습니다.

- 제 말과 AI의 말은 구분된 채로 남습니다.
- 따라서 아이디어의 기원을 되찾고, 제 의도와 시스템의 제안을 구별할 수 있습니다.

세션은 정돈되지 않을 수 있습니다.

- 완벽한 요약을 보여 주는 것이 아니라 생각이 실제로 움직인 과정을 포착하는 데 목적이 있습니다.

컨텍스트가 닫혀 있을 때는 우편함이 말을 전한다

세션 밖에서, 기차 안이나 회의 도중에 인터넷도 없는 상태로 아이디어가 떠오릅니다.

- 그러면 우편함에 넣습니다.
- 이 우편함은 컨텍스트의 루트에 있는 `offline.md` 파일입니다.
- 이 마크다운 파일에 적기만 하면 됩니다.

다음과 같은 형태입니다.

```
현장 기술자가 일반적인 값으로 필수 입력란을 우회하는지 물어볼 것.
```

다음번에 컨텍스트를 열면 이 메모를 다시 읽고, 진행 중인 작업과 연결한 뒤, 알맞은 곳으로 옮깁니다. 할 일이나 인터뷰 질문, 노트가 될 수도 있고, 더는 관련이 없다면 사라질 수도 있습니다.

추가하는 것이 곧 이해하는 것은 아니다

재료가 들어오고 나면 폴더 구조에는 여러 출처와 현재 세션의 기록이 담깁니다.

```
03-sources/
├── documents/
│   ├── SRC00001-entretien-exploitation.md
│   ├── SRC00002-entretien-exploitation.md
│   └── SRC00003-livre-blanc-donnees-intervention.md
└── sessions/
    └── SESSION-2026-09-11.md
```

이 파일들은 재료가 있다는 사실을 입증합니다. 그 재료를 이해했다는 사실까지 입증하지는 않습니다.

`03-sources/ingestion-log.md` 파일은 이 두 작업을 구분합니다. 각 출처의 분석이 완료됐는지, 일부만 끝났는지, 다시 검토해야 하는지를 콘텐츠의 지문과 간단한 처리 메모와 함께 표시합니다.

문서가 교체되거나 텍스트 추출 결과가 달라지거나 세션이 중단될 때 이 구분은 매우 중요해집니다. 컨텍스트는 모든 내용을 이미 안다고 가정하지 않고 분석을 다시 이어 갈 수 있습니다.

재료는 들어왔습니다. 이제 서로 연결하고, 논의하고, 재사용할 수 있을 만큼 명확한 아이디어로 바꿔야 합니다.

기억할 점 – 출처

문서, 세션에서 나온 말, 세션 밖에서 적은 메모는 서로 다른 세 문을 통해 들어오며 각각 원래 형태를 유지합니다. 출처는 재료와 근거를 보존할 뿐, 그로부터 끌어낼 해석을 보존하지는 않습니다. 추가하는 것이 곧 이해하는 것은 아닙니다. 수집 상태 기록은 존재하는 것과 이해한 것을 구분합니다.

5.4

변환하기: 출처에서 아이디어로

이제 제 앞에는 세 개의 출처가 있습니다.

- 인터뷰 두 건은 양식 때문에 생기는 불편을 설명하고,
- 백서 하나는 구조화된 데이터가 필요하다고 주장합니다.

당장 결론을 내려 필수 입력란이 품질을 떨어뜨린다고 쓰고 싶을 수 있습니다. 그러나 너무 성급한 판단입니다. 컨텍스트는 서로 다른 내용을 하나의 결론에 녹여 버리지 않고 보존해야 합니다.

그래서 노트를 만들고 갱신하는 스킬을 호출합니다. 아직 처리하지 않은 내용을 읽고 아이디어를 추출하는 스킬입니다.

홀로 설 수 있는 아이디어를 추출한다

출처 안에 곧바로 재사용할 수 있는 형태의 아이디어가 들어 있는 경우는 드뭅니다.

- 인터뷰에는 사실, 사례, 의견, 망설임, 그리고 그 상황에만 해당하는 세부 사항이 섞여 있습니다.
- 이를 변환한다는 것은 전부 요약하는 것이 아니라 아이디어를 뽑아내는 일입니다.

세 단계를 거칩니다.

```
증언                  "양식이 너무 길면 일부 현장 기술자는 작업을 끝내기
        |                 위해 일반적인 값을 입력합니다."
        ↓
관찰                  입력을 빨리 끝내기 위해 일반적인 값을 사용하는
        |                 경우가 있습니다.
        ↓
안정된 아이디어   지나치게 제약이 많은 입력 방식은 작업 보고서의
                                                 내용을 빈약하게 만들 수 있습니다.
```

강한 아이디어란 눈길을 끄는 문구나 영원한 진리가 아닙니다.

- 분명하고 단단한 중심을 갖고,

- 아이디어가 태어난 원문 밖에서도 이해할 수 있으며,
- 확인하거나 뉘앙스를 더하거나 반박할 수 있을 만큼 범위가 명확해야 합니다.
- 또한 특정한 증언 하나가 일반 법칙으로 바뀌지 않도록 출처와 한계를 함께 보존합니다.

AI는 임무와 지침에 비추어 출처를 읽은 뒤, 이미 있는 노트와 대조합니다. 새로운 아이디어라면 노트를 만들고, 이미 있다면 새로 발견한 뉘앙스나 한계를 기존 노트에 더합니다.

이때 모든 표현을 저에게 일일이 확인받지는 않습니다. 저는 나중에 노트를 다시 읽고 고칠 수 있습니다.

- 세션 중에 모호한 점이 생기면 AI가 제게 묻습니다.
- 나중에 해결할 수 없는 모순을 발견하면 저 대신 결론을 내리지 않고 그대로 보존합니다.


```
04-notes/
└── atomic/
    └── ATO00001-la-contrainte-de-saisie-peut-appauvrir-les-comptes-rendus.md
```

```
---
id: ATO00001
status: active
sources: [SRC00001, SRC00002]
---

# 지나치게 제약이 많은 입력 방식은 작업 보고서의 내용을 빈약하게 만들 수 있다

## 핵심 아이디어

양식 때문에 작업이 늦어지면 현장 기술자는 일반적인 값을 입력하거나 설명을 요구되는 최소한으로 줄일
수 있다.

## 중요한 이유

형식상 구조화된 데이터가 깔끔해 보여도 현장의 현실은 오히려 더 부정확하게 묘사할 수 있다.

## 뉘앙스와 한계

두 인터뷰만으로는 이 현상이 얼마나 자주 발생하는지 측정할 수 없다.

## 출처 / 근거

- SRC00001: 더 빨리 끝내기 위해 일반적인 값을 사용한 사례.
- SRC00002: 필수 입력란을 추가한 뒤 정확성이 떨어진 것으로 관찰됨.

## 연결

- ATO00002와 대조됨.
```

제목은 단순한 주제가 아니라 하나의 명제를 담습니다. 출처 문서를 다시 열지 않아도 노트를 이해할 수 있으며, 동시에 그 기원을 확인할 수 있는 연결도 유지합니다.

원자 노트는 처음 작성된 상태로 고정되지 않습니다.

- 정체성을 잃지 않으면서 새로운 뉘앙스나 근거, 관계를 더할 수 있습니다.

백서에서는 두 번째 노트가 나옵니다. 데이터를 비교하고 종합하고 자동으로 처리하려면 구조화된 데이터가 필요하다는 내용입니다.

- 이 아이디어는 다른 내용을 주장하므로 첫 번째와 분리된 채로 남습니다.

너무 일찍 결론 내리기를 거부한다

두 아이디어는 서로 충돌하는 것처럼 보이지만, 어느 쪽을 택하기에는 아직 자료가 부족합니다. 그래서 모순 항목을 하나 만듭니다.

```
## C-001 – 입력 품질을 떨어뜨리지 않으면서 구조화하기

- 한쪽: 정해진 입력란 때문에 현장 설명이 빈약해질 수 있다.
- 다른 쪽: 대규모로 데이터를 활용하려면 서로 비교할 수 있어야 한다.
- 쟁점: 어떤 정보를 입력 시점에 구조화하고, 어떤 정보를 나중에 구조화할 수 있는지 판단해야 한다.
- 시작일: 2026-09-11
```

모순을 보존하면 서로 대칭적인 두 가지 오류를 피할 수 있습니다. 너무 일찍 한쪽을 선택하거나, 누구도 불편해하지 않을 모호한 표현으로 의견 차이를 덮는 오류입니다.

- 결정이나 새로운 출처, 재정의로 해결될 때까지 열린 채로 남습니다.

모든 불확실성을 이렇게 다룰 필요는 없습니다.

- 이후의 추론이나 결정의 질에 실제로 영향을 주는 긴장에만 이 방식을 사용합니다.

아이디어들이 함께 무엇을 말하는지 연결한다

출처를 받아들이는 과정에서 AI는 원자 노트만 만들지 않습니다.

- 추출한 아이디어와 기존 노트 사이의 관계도 살펴봅니다.
- 여러 아이디어가 하나의 명제로 줄일 수 없는 묶음을 이루면, 같은 작업 안에서 주제 노트를 만들고 서로를 잇는 연결을 추가합니다.

두 번째 명령을 내릴 필요는 없습니다.

- 원자 노트를 만들고, 서로 연결하고, 필요하면 주제 노트를 만드는 과정은 모두 같은 출처 수집 작업에 포함됩니다.

```
---
id: THM00001
status: active
sources: [ATO00001, ATO00002]
---

# 품질은 입력과 활용 모두에 달려 있다

## 관점

작업 보고서의 품질을 채워진 입력란의 수만으로 평가할 수는 없다.

## 종합

시스템은 입력을 부자연스럽거나 힘들게 만들지 않으면서 데이터를 활용할 수 있을 만큼 충분한 구조를 확보해야 한다.

## 연결된 노트

- ATO00001 — 지나치게 제약이 많은 입력 방식은 작업 보고서의 내용을 빈약하게 만들 수 있다.
- ATO00002 — 데이터를 활용하려면 구조화된 데이터가 필요하다.

## 긴장

- C-001 — 입력 품질을 떨어뜨리지 않으면서 구조화하기.
```

주제 노트는 요약을 나란히 놓는 데 그치지 않습니다. 아이디어 사이의 관계를 해석하고 아직 해결되지 않은 것을 보여 줍니다.

같은 단어의 뜻이 사람마다 다르다는 것을 깨닫는다

의견 차이의 일부는 때때로 어휘에서 비롯됩니다. 그럴 때 용어집은 이 컨텍스트에서 사용할 의미를 정합니다.

```
## 구조화된 데이터

비교하거나 자동으로 처리할 수 있도록 명시적인 형식에 따라 기록된 정보.

다음과 혼동하지 말 것: 작업 시점에 사용자가 반드시 채워야 하는 입력란.

예: 고장 유형, 정지 시간, 관련 설비.
```

이 지역적인 정의를 보편적인 정의로 만들려는 것은 아닙니다.

- 두 노트나 두 사람이 같은 단어로 서로 다른 것을 가리키는 일만 막습니다.

결과물에 종속되지 않은 벽돌

노트가 독자에게 어떤 효과를 내기보다 지식을 안정된 형태로 담으려 하는 동안에는 *아이디어의 세계*에 속합니다.

- 따라서 너무 일찍 글의 도입 문구나 슬라이드, 영업 논거, 명세의 일부가 되어서는 안 됩니다.

이런 중립성이 노트를 모호하게 둔다는 뜻은 아닙니다.

- 오히려 제목은 정확한 명제를 주장하고, 본문은 그 뉘앙스를 밝히며, 연결은 토대를 확인할 수 있게 합니다.
- 중립적인 것은 형식입니다. 오직 하나의 결과물에만 쓰이도록 구성되지 않습니다.

따라서 `ATO00001`은 인터뷰 가이드에서 우회 입력에 관한 질문이 될 수도 있고, 검토 문서의 위험 요소나 팀 발표의 사례가 될 수도 있습니다. `ATO00002`는 데이터 활용에 관한 질문을 뒷받침하거나, 제품 요구사항의 근거가 되거나, 자유 입력만으로는 늘 충분하지는 않은 이유를 팀에 설명할 수 있습니다.

주제 노트는 이 벽돌들을 하나의 최종 텍스트로 합치지 않습니다.

- 함께 무엇을 이해하게 하는지 보여 주고, 결과물마다 다르게 다뤄야 할 긴장을 보존합니다.

이 중간 계층 덕분에 결과물을 만들 때마다 분석을 처음부터 다시 하지 않아도 됩니다.

- 출처는 근거로 남고,
- 노트는 활용할 수 있는 지식이 되며,
- 각 결과물은 자신의 필요에 맞게 이를 구성합니다.

모든 아이디어에서 그 근거로 거슬러 올라갈 수 있다

이 단계에서 관계를 나타내면 다음과 같습니다.

```
SRC00001 ┐
         ├──> ATO00001 ┐
SRC00002 ┘             ├──> THM00001
                       ┘
SRC00003 ──> ATO00002 ┘
```

관계에 붙은 이름을 통해 어떤 출처가 아이디어를 확인하고, 보완하고, 반박하는지 구분할 수 있습니다.

- 신뢰 수준을 보면 여러 관찰에 근거한 내용과 아직 가설인 내용을 구별할 수 있습니다.

출처 수집 범위에는 어떤 출처가 실제로 노트에 기여했는지 나타납니다. 폴더에 들어 있기만 한 문서를 이미 이해한 것으로 간주하지 않게 해 줍니다.

이 노트들은 특정 결과물을 위해 작성되지 않습니다.

- 여러 결과물에 활용할 수 있는 공통 지식을 이루며, 어느 하나에 복사되어 갇히지 않습니다.

바로 이처럼 재사용할 수 있는 자산이 *아이디어의 세계*입니다. 다시 찾을 수 있을 만큼 안정적이고, 새롭게 조합할 수 있을 만큼 독립적인 벽돌의 집합입니다.

기억할 점 – 원자 노트

안정된 아이디어는 주제가 아니라 명제를 담으며, 출처 문서를 다시 열지 않아도 이해할 수 있습니다. 추론에 영향을 주는 모순은 너무 일찍 결론 내리거나 모호하게 덮지 않고 그대로 보존합니다. 모든 노트는 출처와의 연결을 유지하므로 아이디어에서 근거로 거슬러 올라갈 수 있습니다.

5.5

결과물을 만들고 결정하기

주제 노트를 통해 정확한 질문 하나가 드러났습니다. 작업이 끝날 때 어떤 정보를 입력해야 하며, 어떤 정보는 나중에 보완해도 될까요?

이제 세 사람이 제게 무언가를 기다립니다.

- 제가 인터뷰해야 할 현장 기술자,
- 결정을 내려야 할 위원회,
- 제가 설명해야 할 팀입니다.

세 번 조사하지는 않을 것입니다. 같은 노트에서 세 번 꺼내 쓸 것입니다.

이때만큼은 어떤 스킬도 세션에서 저를 돕지 않습니다. 결과물을 만드는 일은 되풀이되는 동작이 아니라 글쓰기이기 때문입니다.

같은 자산에서 세 번 꺼내 쓴다

노트를 결과물로 옮기는 일은 복사가 아닙니다. 각 결과물은 자신에게 유용한 벽돌을 선택하고, 받을 사람에 맞춰 순서를 정하고 다시 표현합니다.

공통 벽돌	인터뷰 가이드	검토 문서	팀 발표 자료
AT000001 — 지나치게 제약이 많은 입력 방식은 작업 보고서의 내용을 빈약하게 만들 수 있다	어떤 입력란을 어떻게 우회하는지 묻기	품질 저하 위험 설명하기	현장의 불편을 보여 주는 사례 제시하기
AT000002 — 데이터를 활용하려면 구조화된	실제로 어떤 데이터를 다시 사용하는지 묻기	비교와 자동화의 필요성 밝히기	팀이 얻게 될 가치를 설명하기

공통 벽돌	인터뷰 가이드	검토 문서	팀 발표 자료
데이터가 필요하다			
THM00001 — 품질은 입력과 활용 모두에 달려 있다	두 가지 요구를 중심으로 인터뷰 구성하기	쟁점과 판단 기준 표현하기	이야기의 흐름 구성하기
C-001 — 입력 품질을 떨어뜨리지 않으면서 구조화하기	긴장을 확인할 질문으로 바꾸기	해결해야 할 문제로 보존하기	성급한 해결책을 발표하지 않고 드러내기

이처럼 같은 자산으로 세 가지 구성을 만들지만, 어느 결과물도 모든 내용을 가져가거나 그 바탕이 된 노트를 대체하지 않습니다.

이 구분의 가장 큰 장점은 지식을 한 번만 바로잡으면 된다는 것입니다.

- 새로운 출처가 ATO00001에 뉘앙스를 더하면, 다음 결과물들은 여러 문서에 흩어진 예전 표현을 되풀이하지 않고 더 풍부해진 이 벽돌에서 다시 출발합니다.

가장 급한 다음 인터뷰부터 시작한다

가장 먼저 필요한 것은 현장 기술자와의 인터뷰를 준비하는 일입니다. 결과물은 초안 영역에서 시작됩니다.

```
05-outputs/
└── _draft/
    └── questions-prochain-entretien.md
```

```
# 다음 인터뷰를 위한 질문

1. 작업 중에 가장 시간을 지체시키는 입력란은 무엇입니까?
2. 요구받은 정보를 알 수 없을 때는 무엇을 입력합니까?
3. 작업이 끝난 뒤 추가할 수 있는 정보는 무엇입니까?
4. 운영에 활용하려면 어떤 데이터가 처음부터 정확해야 합니까?
```

노트는 재사용할 수 있는 아이디어를 보존하려 합니다. 반면 결과물은 받을 사람과 형식, 기대하는 행동을 선택합니다.

다시 읽고 승인한 파일은 `_draft/`를 떠나 `05-outputs/`로 바로 이동합니다. 이 이동은 문서의 종류가 아니라 완성도를 나타냅니다.

지금까지는 결과물을 만들기만 했습니다. 이미 알고 있는 내용을 세 부류의 독자에 맞게 표현했습니다. 이 장의 다음 부분에서는 전혀 다른 행동을 다룹니다. 어느 순간에는 이미 아는 것을 되풀이하는 일을 멈추고 결정을 내려야 합니다. 그리고 그 결정은 6개월 뒤에도 찾을 수 있어야 합니다.

결정 기록은 결과를 빚어내는 선택을 보존한다

"명세를 작성하기 전에 현장 기술자를 인터뷰한다"는 말은 무엇을 제공할지에 관한 판단이 아닙니다.

- 단순히 다음 행동이라면 `01-state/todo.md`에 들어갑니다.
- 모든 탐색 작업이 따라야 할 규칙이라면 `00-frame/directives.md`의 지침이 됩니다.

어느 경우에도 결정 기록이 필요하지는 않습니다.

결정 기록은 작업 과정에서 여러 방향이 드러나고, 그중에서 제품이나 글, 방법 또는 컨텍스트의 다른 결과를 좌우할 하나를 골라야 할 때 등장합니다. 무엇을 설계하거나 제공하기로 했는지, 배제한 대안은 무엇인지, 왜 그렇게 선택했는지, 어떤 결과를 감수하기로 했는지를 보존합니다.

따라서 이 사례에서 판단할 대상은 인터뷰를 어떻게 조직할지가 아닙니다. 기능에 관한 결정입니다.

- 작업이 끝날 때 모든 정보를 입력하게 할 것인가,
- 자유 입력만 유지할 것인가,

- 아니면 작업 보고서를 나중에 한 번 더 보완할 수 있게 할 것인가입니다.

유용한 결정 기록은 선택의 핵심을 구현하는 사람에게 떠넘기지 않습니다. 세 사람이 읽더라도 같은 범위와 예상 동작, 한계를 이해해야 합니다. 구조를 좌우하지 않는 기술적 세부 사항은 열어 둘 수 있지만, 그 어느 것도 결정을 소리 없이 바꿀 수 있어서는 안 됩니다.

```
04-notes/
└── decision-records/
    └── DRE00001-limiter-la-cloture-a-quatre-informations.md
```

작성하고 나면 다음과 같은 모습입니다. 이 템플릿이 긴 이유는 결정을 이해하고 나중에 다시 검토할 수 있게 하는 추론 전체를 보존하기 때문입니다. 결정 기록에 이미 익숙한 독자라면 곧바로 뒤의 설명으로 넘어가도 됩니다.

```
---
uuid: 0d5efbe2-1eac-4ad7-94c1-29f68adcf001
id: DRE00001
created: 2026-09-11
updated: 2026-09-11
status: accepted
decided_by: 제품 팀
decided_on: 2026-09-11
supersedes: null
superseded_by: null
sources: [ATO00001, ATO00002, THM00001]
---

# 작업 완료에 필요한 필수 정보를 네 가지로 제한한다

## 결정

작업을 완료된 것으로 처리하려면 정확히 다음 네 가지 정보만 갖춰져 있으면 된다.

1. 관련 설비. 작업 카드에 이미 표시되어 있거나 완료 시점에 입력해야 한다.
2. 작업이 실제로 끝난 날짜와 시간.
3. 작업 종료 시점의 설비 상태. 정상 가동으로 복귀, 제한 운전 상태로 유지, 또는 가동 중단 상태로
남음
  가운데 하나.
4. 수행한 작업 또는 설비를 다시 가동하지 못한 이유를 설명하는, 비어 있지 않은 요약.

그 밖의 어떤 정보도 작업 완료를 막지 않는다.

작업 완료 뒤에도 고장 유형, 확인된 원인, 수행한 작업의 종류, 교체한 부품, 설비를 사용할 수 없었던
```

```
시간을
  작업 보고서에 추가할 수 있으며, 작업의 완료 상태는 바뀌지 않는다.
현장 기술자가 이 추가 정보를 입력할 수 있지만, 운영 책임자는 24시간 안에 이를 보완하거나 승인해야
한다.

아직 미완성인 작업 보고서도 열람할 수 있다.
'보완 필요'로 표시하고, 24시간이 지나면 운영 책임자에게 알린다.

## 맥락

현장 기술자는 분석용 전체 양식을 훑지 않고도 현장에서 작업을 완료할 수 있어야 한다.
그러나 운영 부서에서 고장과 그 원인, 수행한 작업, 교체한 부품, 설비를 사용할 수 없었던 시간을 분
석하려면 서로
  비교할 수 있는 정보가 필요하다.

## 가정

- 작업을 시작하기 전에 작업 카드에서 설비를 식별하는 것이 일반적이다.
- 현장 기술자는 작업 완료 시점에 설비의 상태를 알고 있으며 수행한 작업을 간략히 설명할 수 있다.
- 운영 책임자는 24시간 안에 완전한 작업 보고서를 승인할 시간과 필요한 정보를 갖고 있다.

## 대안

- 완료 시점에 아홉 가지 정보를 모두 입력하게 한다.
- 자유로운 요약문만 보존하고 나중에 작업 정보를 자동으로 추출한다.
- 모든 정보를 선택 사항으로 두고 나중에 품질만 점검한다.

## 이유

즉시 입력하는 네 가지 정보만으로 작업을 식별하고, 종료 시각을 기록하고, 설비 상태를 파악하고, 수행
한 작업을
  이해할 수 있다.
분석용 정보 다섯 가지도 필요하지만, 그 때문에 현장 기술자가 현장에서 작업을 끝내지 못해서는 안 된
다.
두 번째 단계에서는 작업 보고서를 보완하거나 승인할 책임이 운영 책임자에게 있음을 명시한다.

## 근거

- ATO00001은 입력 제약 때문에 생기는 우회 행동을 설명한다.
- ATO00002는 서로 비교할 수 있는 데이터의 필요성을 확립한다.
- THM00001은 두 요구를 연결한다.

## 결과

### 긍정적인 결과

- 작업을 완료할 때는 즉시 알 수 있는 네 가지 정보만 요구한다.
- 작업 보고서를 보완한 뒤에는 작업 분석에 필요한 정보를 이용할 수 있다.
- 빠진 데이터의 책임자가 분명하게 드러난다.

### 감수하는 조건

- 작업 지표는 작업 뒤 24시간 동안 불완전할 수 있다.
- 운영 책임자에게 추가 승인 업무가 생긴다.

### 위험
```

```
- 미완성 작업 보고서가 쌓일 수 있다.
- 나중에 추가한 정보는 현장에서 확인한 상황을 덜 충실하게 반영할 수 있다.

## 준수 기준

- 네 가지 즉시 입력 정보 가운데 하나라도 빠지면 작업을 완료 처리할 수 없다.
- 다섯 가지 추가 정보가 하나도 없어도 작업을 완료 처리할 수 있다.
- 추가 정보는 작업의 완료 상태를 바꾸지 않고 수정할 수 있다.
- 작업 보고서는 사용되는 모든 곳에서 '보완 필요' 또는 '완료 및 승인됨'으로 표시된다.
- 작업 완료 24시간 뒤에도 작업 보고서가 완성되고 승인되지 않았다면 운영 책임자에게 알린다.

## 관련 결정

- C-001 – 입력 품질을 떨어뜨리지 않으면서 구조화하기를 해결함.

## 출처

- SRC00001
- SRC00002
- SRC00003
```

여기서 accepted 상태는 판단이 승인됐으며 구현을 이끌 만큼 충분히 구체적이라는 뜻입니다.

- 필요한 정보와 책임, 준수 기준이 정해지지 않은 동안에는 결정 기록의 상태를 proposed로 유지해야 합니다.
- 승인된 뒤에는 역사적 기록으로 고정되며 아무도 모르게 바뀌지 않습니다.

새로운 결정이 이를 대체하더라도 이전 결정 기록은 계속 열람할 수 있으며, 두 파일은 supersedes와 superseded_by로 연결됩니다.

이 연속성은 판단의 전체 시간 순서를 보존합니다. 무엇을 어떤 정보에 근거해 결정했으며, 이후 어떤 요소 때문에 다른 선택을 했는지가 남습니다.

따라서 6개월 뒤에는 현재의 규칙만 다시 찾는 것이 아닙니다. 첫 번째 방향이 왜 옳아 보였는지, 무엇 때문에 부족해졌는지, 다음 결정이 어떻게 만들어졌는지 이해할 수 있습니다.

이런 시간의 흐름이 없으면 연이은 선택들이 결국 앞뒤가 맞지 않는 것처럼 보입니다. 흐름을 보존하면 변화하는 제품의 이해할 수 있는 역사가 됩니다.

- 대안은 무엇을 배제했는지 보여 줍니다.
- 감수한 조건은 무엇을 희생했는지 드러냅니다.
- 근거는 결정을 뒷받침하는 지식과 증거에 연결합니다.

결과물을 만들어도 내가 아는 것은 소진되지 않는다

인터뷰 가이드와 주제 노트, 결정 기록은 같은 문제를 다루지만 중복되지 않습니다. 서로 다른 세 가지 기능을 하기 때문입니다.

- 노트는 컨텍스트가 안다고 여기는 것을 보존합니다.
- 결정은 하나의 선택과 그 이유를 보존합니다.
- 가이드는 이 지식을 특정한 상대를 위한 행동으로 바꿉니다.

다른 결과물도 같은 공통 지식을 다른 형태로 다시 사용할 수 있습니다. 결과물을 만든다고 해서 **컨텍스트**의 내용이 소진되는 것은 아닙니다. 오히려 그 내용으로 무엇을 할 수 있는지가 드러납니다.

이제 도달한 상태를 기록해야 합니다. 그래야 다음 세션에서 지금까지의 경로를 재구성하지 않고 작업을 이어 갈 수 있습니다.

기억할 점 – 결정 기록

같은 노트가 여러 결과물에 쓰이며, 어떤 결과물도 자신이 사용하는 지식을 소모하지 않습니다. 결정 기록은 해야 할 행동이 아니라 무엇을 설계하거나 제공할지를 좌우하는 판단을 보존합니다. 무엇을 배제하고 무엇을 희생했는지 보존하므로 나중에 선택을 다시 검토할 수 있습니다.

5.6

저장하고 재개하기

결정 기록이 방금 승인됐고 회의는 3분 뒤 시작됩니다. 조사를 끝낸 것도 아니고, 끝처럼 보이는 단계 하나를 마친 것도 아닙니다.

흔히 있는 일입니다. 다른 업무에 주의를 돌려야 하거나, 컴퓨터를 다시 시작해야 하거나, 다른 컴퓨터로 옮겨 가거나, AI의 컨텍스트 창이 너무 가득 찰 수 있습니다.

- 따라서 저장은 "조사가 끝났다"와 같이 작업 자체의 필요 때문에 하는 일이 아닙니다.
- 기술적인 보장입니다. 컨텍스트의 상태가 그것을 만든 세션이 끝난 뒤에도 살아남아야 합니다.

지금 멈췄다가 나중에 다시 열어도, 작업을 두고 간 바로 그 지점에서 되찾을 수 있어야 합니다.

스킬 하나가 이 일을 맡으며 몇 초 안에 끝나야 합니다. 저장 비용이 크면 서두를 때, 다시 말해 저장이 가장 필요한 순간에 실행하지 않게 되기 때문입니다.

AI가 그 시점까지 우리가 아는 내용을 고정한다

`01-state/state.md` 파일은 다음 작업의 방향을 정할 만큼 안정된 내용을 요약합니다.

```
contexte-qualite-comptes-rendus/
└── 01-state/
    └── state.md
```

저장하는 순간에는 다음과 같은 형태가 될 수 있습니다.

```
# 현재 상태

## 안정된 지식

- 지나치게 제약이 많은 입력 방식은 작업 보고서의 내용을 빈약하게 만들 수 있다.
- 대규모로 데이터를 활용하려면 그래도 구조화된 데이터가 필요하다.

## 확정된 결정

- 작업을 완료하려면 관련 설비, 종료 날짜와 시간, 설비 상태, 수행한 작업의 요약이 필요하다.
- 고장 유형과 원인, 작업 종류, 교체 부품, 설비를 사용할 수 없었던 시간은 작업 완료 뒤에 보완하며, 운영
  책임자가 24시간 안에 승인한다.

## 핵심 불확실성

- 이러한 분리가 데이터 품질을 떨어뜨리지 않으면서 실제로 작업 완료 시간을 줄이는지 측정해야 한다.

## 지식 부채

- 작업 완료 시간이나 24시간 뒤 완성된 작업 보고서의 비율을 측정한 시범 운영은 아직 없다.
```

- 완료된 할 일은 할 일 목록에서 사라질 수 있습니다.
- 해결되지 않은 모순은 상태에 계속 드러납니다.

종합에는 작업을 통해 충분히 안정된 내용만 들어갑니다. 상태는 현재를 설명하도록 다시 작성되고, 종합은 지금까지 얻은 이해를 보존하도록 점차 내용이 늘어납니다.

미래의 나를 위해 단 하나의 행동을 준비한다

작업 재개 파일은 전체 역사를 이야기하지 않습니다. 다음에 할 유용한 행동을 준비합니다.

```
contexte-qualite-comptes-rendus/
└── 01-state/
    └── handoff.md
```

`01-state/handoff.md` 파일은 다음과 같은 형태가 될 수 있습니다.

```
# 작업 재개

## 현재 상태

DRE00001이 승인됐고 C-001은 해결됐다.
작업 완료에 필요한 정보 네 가지와 추가 정보 다섯 가지가 확정됐다.

## 주의할 점

DRE00001을 대체하는 새로운 결정 기록을 열지 않고 작업 완료 시점의 필수 정보를 추가해서는 안 된다.

## 다음 행동

DRE00001 구현을 시작한다.

## 열린 항목

- 작업 뒤 설비가 가질 수 있는 세 가지 상태를 현장 기술자에게 어떤 이름으로 보여 줄지 정한다.
- 작업 완료 시간과 24시간 뒤 완성된 작업 보고서의 비율을 측정할 방법을 준비한다.
```

유용한 작업 재개 정보는 다음 행동을 하나만 제시합니다. 열 가지 가능성을 나열하면 다음 세션은 제가 방금 마친 결정 작업을 다시 해야 합니다.

작업을 다시 시작할 때 비교할 흔적을 남긴다

`01-state/checkpoints.md` 파일은 현재 상황을 압축한 사진을 차례로 보존합니다.

```
contexte-qualite-comptes-rendus/
└── 01-state/
    └── checkpoints.md
```

파일은 다음과 같은 형태가 될 수 있습니다.

```
## 2026-09-11

- 출처: 3
- 원자 노트: 2
- 주제 노트: 1
- 열린 모순: 0
- 승인된 결정 기록: 1
- 다음 행동: DRE00001 구현 시작.

## 2026-09-04

- 출처: 1
- 원자 노트: 1
- 주제 노트: 0
- 열린 모순: 0
- 다음 행동: 운영 부서의 관점 확인.
```

이 사진들은 모든 파일을 비교하지 않고도 컨텍스트가 어떻게 변했는지 보여 줍니다. 또 제가 알아차리지 못한 채 작업이 정체되거나 부채가 늘어나는 것도 발견하게 해 줍니다.

AI는 자신의 생각이 아니라 일어난 일을 기록한다

이력은 모든 대화를 재구성하려 하지 않고 중요한 사건을 기록합니다. 이 사례에서는 그날 일어난 사건이 `02-history/2026-09-11.md`에 추가됩니다.

```
contexte-qualite-comptes-rendus/
└── 02-history/
    └── 2026-09-11.md
```

추가되는 부분은 다음과 같은 형태입니다.

```
## 2026-09-11

- SRC00003-livre-blanc-donnees-intervention 추가.
- ATO00002와 THM00001 생성.
- C-001 시작.
- `questions-prochain-entretien.md` 승인.
- DRE00001 승인 및 C-001 해결.
```

- 상태는 현재 어디까지 왔는지 말합니다.
- 이력은 무슨 일이 있었는지 말합니다.
- 작업 재개 정보는 다음에 무엇을 해야 하는지 말합니다.

내가 보지 못할 것을 시스템이 확인하게 한다

저장할 때 몇 가지 기계적인 검사를 하면 누락을 줄일 수 있습니다. 결과는 `01-state/health.md`에 기록됩니다.

```
contexte-qualite-comptes-rendus/
└── 01-state/
    └── health.md
```

- 참조한 파일이 모두 존재하는가?
- 식별자는 고유한가?
- 모든 노트에 출처가 표시되어 있는가?
- 추가했지만 분석하지 않은 출처가 표시되어 있는가?
- 대체된 결정이 그 결정을 대신한 새 결정과 연결되어 있는가?
- 다음 행동이 명시되어 있는가?

이 검사가 제 판단을 대신하지는 않습니다. 구조상의 오류가 생각의 문제인 것처럼 보이는 일을 막을 뿐입니다.

전체 순환과 그것이 가능하게 하는 것

이제 전체 흐름이 보입니다.

```
열기
    ↓
우편함 읽기
    ↓
임무, 상태, 작업 재개 정보 불러오기
    ↓
출처 추가
    ↓
노트 만들기 또는 갱신
    ↓
초안 만들기
    ↓
점검, 승인 또는 결정
    ↓
상태, 종합, 작업 재개 정보 갱신
    ↓
체크포인트와 이력
    ↓
저장 점검
    ↓
다음번 열기
```

저장하기 위해 이 흐름이 끝날 때까지 기다리지는 않습니다.

- 출처를 추가한 뒤, 노트를 만드는 도중, 결정을 내리기 전, 결과물을 준비하는 중에 저장합니다. 컨텍스트 창이 터질 듯 가득 찬 것이 보일 때도 마찬가지입니다.

바로 이것이 저장의 역할입니다. 현재 대화와 상관없이 각 상태에서 작업을 재개할 수 있게 하는 것입니다.

이제 회의를 시작해도 됩니다.

컨텍스트는 단순한 파일 폴더 구조가 아닙니다. 재료를 받아들이고, 긴장을 보존하고, 아이디어를 안정된 형태로 만들고, 여러 결과물을 생산하며, 작업 재개를 준비함으로써 긴 시간에 걸쳐 생각하게 해 주는 시스템입니다.

형태는 의도적으로 단순하고 다른 도구로 옮기기 쉽게 유지했습니다. 읽을 수 있는 파일만으로도 작업을 관찰하고, 검증하고, 전달할 수 있습니다.

이 엔진을 이해하고 나면 또 다른 질문이 생깁니다. **컨텍스트**는 자신의 바깥에 존재하는 도구, 지식 기반, 실제 제품과 어떻게 대화할까요?

기억할 점 – 저장

저장은 한 단계의 끝을 표시하는 일이 아닙니다. 상태를 만든 세션이 끝난 뒤에도 그 상태가 살아남도록 보장하는 일입니다. 상태는 현재 어디까지 왔는지, 이력은 무슨 일이 있었는지, 작업 재개 정보는 다음에 무엇을 해야 하는지 말합니다. 이 순환은 어느 지점에서든 중단할 수 있으며, 모든 지점에서 작업을 재개할 수 있습니다.

부 6

생태계 안의 컨텍스트

잘 관리된 컨텍스트를 보면 이용할 수 있는 모든 지식을 그 안에 넣고 싶어집니다.

이 부에서는 컨텍스트가 밖에 남겨 두는 것과 그 이유를 살펴봅니다. 생각할 수 있는 상태를 유지하기 위해 규모를 작게 유지합니다. 혹시 몰라 보관한 정보는, 원본이 다른 곳에서 계속 바뀌는 동안 알림도 없이 낡아 버립니다. 그래서 보관과 측정, 실행을 전문으로 하는 도구, 그리고 서로 다른 질문을 탐구하는 다른 컨텍스트와 협업합니다.

6.1

구조화된 지식에 질문하기

작업 보고서는 잠시 미뤄 두겠습니다. 오늘 아침에는 같은 제품의 다른 주제인 가입 도중 이탈하는 사람들을 다룹니다. 한 조사 업체가 방금 연구 결과를 보내왔고, 저는 이를 컨텍스트에 추가합니다. 참여자 몇 명이 비밀번호 규칙 앞에서 망설인다는 내용입니다.

이 관찰을 보니 전에 접한 아이디어가 떠오릅니다. 사용자가 제품의 가치를 느끼기 전에 큰 노력을 요구하면 떠날 수 있다는 것입니다.

- 이 주제에 관한 자료가 있다는 것은 거의 확실합니다.
- 다만 어디에 있는지, 당시 어떤 표현을 썼는지 기억나지 않습니다.

초기 노력, 불편, 활성화, 온보딩 중 어떤 말로 기록했을까요? 그 아이디어는 어쩌면

- 책에서 추출됐거나,
- 프로덕트 매니지먼트 노트에서 발전했거나,
- 다른 사용자 여정과 연결됐을 수 있습니다.
- 그 밖의 가능성도 있습니다.

여러 키워드로 검색해 결과를 하나씩 열어 볼 수도 있습니다. 대신 컨텍스트에 묻습니다.

> 사용자가 제품의 가치를 느끼기 전에 요구받는 노력에 관해 우리가 이미 배운 것은 무엇인가?

컨텍스트가 더 넓은 기억에 질문한다

답은 현재 활성화된 컨텍스트 안에 없습니다. 하지만 책과 글, 표준, 그 밖의 문서에서 얻은 수천 개의 노트가 이미 구조화되어 있는 공용 도서관인 제 LLM 위키에는 있을 수 있습니다.

LLM 위키와 컨텍스트는 성격이 다릅니다.

위키는 지식 기반입니다.

- 다른 사람이 쓴 내용에서 정의, 원칙, 연구 결과, 표준 요구사항을 뽑아 구조화된 노트로 보존합니다. 제 경우에는 원자 노트 개념을 재사용했습니다.
- 각 노트는 알려진 내용과 그 출처를 밝힙니다.
- 수년에 걸쳐 서로 아무 관련이 없는 주제까지 쌓아 온 수만 개의 노트를 담을 수 있습니다.

어떤 컨텍스트도 이 정도 양을 담을 수 없으며, 그것은 컨텍스트의 역할도 아닙니다.

그러나 위키는 거기서 멈춥니다.

- 그 지식이 오늘 제가 하는 작업에서 왜 중요한지, 제가 그것으로 무엇을 했는지는 말해 주지 않습니다.
- 아이디어를 결정과 연결하거나, 결정을 그 대가와 연결하지 않습니다.

LLM 위키는 사고 시스템이 아닙니다. 대량으로 보존하지만 추론하지는 않습니다.

추론은 컨텍스트 안에 존재합니다. 원인과 과정을 보존하는 쪽은 컨텍스트입니다.

- 임무는 무엇인지,
- 어떤 지침과 결정이 확정됐는지,
- 이 임무를 위해 왜 이 아이디어를 선택했는지,
- 어떤 대안을 두고 판단했는지,
- 무엇을 포기하기로 했는지,
- 어떤 모순이 아직 열려 있는지를 보존합니다.

사람이 다시 읽을 수 있는 수십 개의 노트 정도로 작게 유지하기에 가능한 일입니다.

둘 다 자신의 강점에 대한 대가를 치릅니다.

- 위키는 규모를 얻는 대신 흐름을 잃습니다.

- **컨텍스트**는 흐름을 지키는 대신 규모를 포기합니다.

바로 그렇기 때문에 둘은 합쳐지지 않고 서로 대화해야 합니다.

찾고 있는 아이디어를 설명한다

가입 문제를 다루는 컨텍스트 안으로 위키 전체를 복사하지 않습니다. RAG를 한 번 실행한 뒤 그래프에서 다시 검색하지도 않습니다.

그저 컨텍스트에 질문합니다.

> 사용자가 제품의 가치를 느끼기 전에 요구받는 노력에 관해 우리가 이미 배운 것은 무엇인가?

제 말로 질문하면 됩니다.

- 노트 제목이나 위치, 작성 당시 사용한 용어를 알 필요가 없습니다.

컨텍스트는 API를 통해 이 아이디어를 LLM 위키에 전달합니다. 나머지는 모두 위키 안에서 이루어집니다.

위키는 의미로 먼저 찾고, 그 결과의 주변을 살핀다

LLM 위키는 먼저 제 질문과 의미가 가까운 노트를 찾습니다.

- 어떤 노트는 '노력'이라는 단어 없이 초기 불편이나 활성화, 첫 가치를 얻기까지의 시간을 다룰 수 있습니다.

RAG 덕분에 같은 어휘를 요구하지 않고도 제 아이디어를 이미 구조화된 지식과 연결할 수 있습니다.

그다음 위키는 가장 좋은 결과의 주변을 살핍니다. 의미 그래프는 노트 사이에 이미 만들어진 관계를 따라가며 주제를 밝히는 데 도움이 되는 이웃 노트를 더합니다.

- 정의,
- 한계,
- 모순,
- 보완하는 아이디어가 해당됩니다.

이 이웃 노트 가운데 일부는 제 질문에 직접 응답하지 못했을 수도 있습니다. 그러나 아이디어가 어떤 조건에서 적용되고 어느 지점에서 더는 참이 아닌지를 보여 주기 때문에 소중해집니다.

```
내 말로 표현한 아이디어
        |
        ▼
위키가 의미상 가까운 노트를 찾는다
        |
        ▼
유용한 주변 노트를 더한다
        |
        ▼
출처가 있는 참고 자료 몇 개를 돌려준다
        |
        ▼
컨텍스트가 자신의 임무에 필요한 것만 선택한다
```

RAG는 제가 이름 붙일 줄 몰랐던 것을 찾아냅니다. 그래프는 제가 찾아볼 생각조차 못 했던 것을 드러냅니다.

저에게 이 두 작업은 하나로 느껴집니다. 문제에 답하는 참고 자료를 요청하면, LLM 위키가 아이디어와 그 주변을 찾아 짧은 목록으로 돌려줍니다.

컨텍스트는 완성된 답이 아니라 참고 자료를 받는다

위키는 기존 노트를 내용과 출처, 관계와 함께 돌려줍니다. 여러 결과를 남모르게 섞은 종합문을 새로 만들지는 않습니다.

따라서 컨텍스트는 제안된 항목을 하나씩 읽고, 가입 과정에서 관찰한 내용과 비교하며, 근거를 확인해야 할 때 원래 출처로 돌아갈 수 있습니다.

- 어떤 노트는 제품에 접근하기 전에 요구되는 노력을 정확히 다룹니다.
- 다른 노트는 사용자가 제품의 가치를 이미 안 뒤에 요구되는 노력을 다룹니다.

제 질문과 가깝다는 이유만으로 둘이 같은 내용이 되지는 않습니다.

컨텍스트는 자신의 임무에 따라 선별합니다. 다음과 같은 일을 할 수 있습니다.

- 가입 도중의 이탈을 이해하게 해 주는 아이디어를 선택하고,

- 모순을 보존하며,
- 지나치게 일반적인 노트를 제외합니다.

선택된 노트는 그 기원과 근거가 된 출처를 유지한 채 컨텍스트 안으로 들어옵니다. 나머지는 위키에 남습니다.

따라서 모든 컨텍스트를 제 도서관 전체의 사본으로 만들지 않고도 훨씬 더 넓은 기억의 도움을 받을 수 있습니다.

- 이 기억이 어떻게 색인됐는지 배울 필요도 없습니다. 찾는 아이디어를 설명한 뒤, 돌아온 참고 자료를 확인하면 됩니다.

이 연결을 통해 이미 구조화된 지식에 접근할 수 있습니다.

- 그러나 제품이 오늘 실제로 어떻게 작동하는지는 아직 알려 주지 않습니다.

이 다른 질문에 답하려면 컨텍스트가 도서관을 나와 코드와 테스트, 실행 중인 제품에 직접 질문해야 합니다.

기억할 점 – 두 시스템으로 들어가는 하나의 문

LLM 위키는 규모를 얻는 대신 흐름을 잃고, 컨텍스트는 흐름을 지키는 대신 규모를 포기합니다. 제가 아이디어를 제 말로 설명하면, 위키가 의미상 가까운 노트를 찾고 유용한 주변 노트를 더합니다. 컨텍스트는 출처가 있는 참고 자료 몇 개를 받아 자신의 임무에 필요한 것만 선택합니다.

6.2

실제 제품에 질문하기

가입 도중 이탈하는 문제에 관한 연구를 다시 읽고 있는데 질문 하나가 들어옵니다.

> 초대받은 사용자도 다른 사용자처럼 비밀번호를 만들어야 하나요?

아주 작은 질문처럼 보입니다. 하지만 고객에게 답해야 하는 고객 지원 담당자가 기다리고 있습니다.

저는 다음과 같이 할 수도 있습니다.

- 읽던 것을 멈추고,
- 테스트 환경을 열고,
- 계정 두 개를 준비하고,
- 각 사용자 여정을 다시 실행한 뒤,
- 벌어지는 일을 비교하는 것입니다.

예전에는 그렇게 했습니다.

- 30분쯤 지나면 아마 답을 얻었을 것입니다.
- 하지만 하던 작업의 흐름도 잃었을 것입니다.

이번에는 컨텍스트에 질문합니다.

컨텍스트가 제품을 직접 살펴본다

컨텍스트는 사용자 연구에서 나온 노트로 작업하며, 그 주장의 출처로 연구 자료도 보존합니다.

- 현재 임무도 담고 있습니다.
- 질문이 왜 중요한지는 이해하지만, 이런 요소만으로 제품의 현재 동작까지 알 수는 없습니다.

저는 더 이상 제품의 동작을 계속 설명하는 명세를 별도로 관리하지 않습니다.

- 제품이 너무 자주 바뀌므로 옆에 놓인 문서가 계속 충실한 사본으로 남을 수 없기 때문입니다.

언제나 최신인 유일한 진실은 배포되어 운영 환경에서 실행 중인 것, 즉 애플리케이션의 소스 코드입니다.

제 시스템에서는 컨텍스트 외부의 스킬들이 소스 코드에 질문하고 역문서화를 만들 수 있습니다. 컨텍스트는 제품 질문에 답하는 스킬을 호출해 제 요청을 전달합니다.

```
프로덕트 매니저의 질문
          │
          ▼
       컨텍스트
          │  호출
          ▼
제품을 읽는 스킬
          │  질문
          ▼
       실제 제품
```


스킬은 먼저 코드에서 초대 사용자 여정을 찾습니다. 그런 다음 계정 생성 규칙까지 이 여정을 따라갑니다.

코드는 프로그램된 동작을 보여 줍니다. 테스트를 보면 초대받은 사용자는 즉시 비밀번호를 만들지 않고도 받은 링크를 통해 가입을 마칠 수 있습니다. 설정을 확인하면 이 여정에 적용되는 고객별 변형이 활성화되어 있지 않습니다.

스킬은 기능 관점의 답을 돌려줍니다.

> 초대받은 사용자는 이 단계에서 비밀번호를 만들지 않고도 초대 링크를 통해 계정에 접근할 수 있습니다. 나중에 비밀번호를 만들기로 선택하면 일반 규칙이 적용됩니다. 테스트는 초대 사용자 여정과 사용자가

> 원해서 비밀번호를 만드는 과정을 확인합니다. 비밀번호 찾기 과정은 조사하지 않았습니다.

첫 두 문장은 고객 지원팀에 전달할 수 있습니다. 개발자와 함께 분석을 확인하고 싶다면 스킬이 제공한 참조를 열어 볼 수도 있습니다.

마지막으로 이 답을 근거와 확인 날짜, 명시된 한계와 함께 컨텍스트에 넣을 수 있습니다.

- 고객 지원팀에 답하면 하나의 사례를 해결하고 끝납니다.
- 컨텍스트에 축적하면 제품에 관해 확인된 사실이 해당 임무의 *아이디어의 세계*로 들어갑니다. 다음에 초대 사용자 여정을 묻는 질문은 처음부터 다시 시작하지 않으며, 연구에서 나온 노트에는 관찰만으로 알 수 없었던 내용, 즉 각 여정의 어느 시점에 비밀번호를 요구하는지가 더해집니다.

답의 가치는 함께 제시되는 근거에서 나온다

스킬은 그럴듯한 답만 만들어 내지 않았습니다. 각 주장에 대해 무엇을 살펴봤는지 밝힙니다.

- 코드에 있는 규칙,
- 동작을 확인하는 테스트,
- 어떤 변형이 활성화되어 있는지를 정하는 설정입니다.

이 흔적은 확신에 차 있지만 틀린 답을 막아 줍니다. 무엇이 빠졌는지도 즉시 보여 줍니다.

여기서 스킬은 비밀번호 찾기 과정을 확인하지 않았습니다. 읽지 않은 여정까지 결론을 넓히지 않고 그 사실을 밝힙니다.

질문에 따라 다른 요소가 필요할 수도 있습니다.

- 코드에는 가능한 규칙이 들어 있지만, 어떤 규칙이 활성화됐는지는 기능 플래그가 알려 줍니다.
- 데이터베이스 구조는 제품이 무엇을 기록할 수 있는지 보여 주지만, 권한이 허용된 데이터가 있어야 특정 계정의 상황을 이해할 수 있습니다.

- Git 이력과 Pull Request는 규칙이 언제 바뀌었는지, 그 변경과 함께 어떤 설명이 보존됐는지 찾는 데 도움이 됩니다.
- 그 밖의 요소도 있습니다.

스킬은 질문에 답하는 데 필요한 것만 모읍니다. 저장소 전체를 빨아들이거나 모든 파일을 컨텍스트의 출처로 바꾸지 않습니다.

답은 고객 지원팀과 영업팀, 프로덕트 매니저가 이해할 수 있는 형태로 남습니다. 답을 검증하거나 조사를 이어 갈 사람은 근거도 확인할 수 있습니다.

규칙을 찾는 것과 바꾸기로 결정하는 것은 다르다

가입 과정 연구로 돌아갑니다. 이 연구를 통해 두 사용자 여정에서 비밀번호 생성을 서로 다른 시점에 제시한다는 사실을 이미 알았습니다.

스킬은 연구에서 볼 수 없었던 내용을 더합니다. 초대받은 사용자가 비밀번호 규칙을 피해 가는 것은 아닙니다. 더 나중에 마주칠 뿐이며, 그때 적용되는 규칙은 정확히 같습니다.

따라서 이 사용자에게 불편이 사라진 것이 아니라 위치가 바뀐 것입니다. 이 세부 사항 때문에 질문이 달라집니다. 누가 규칙을 겪는지가 아니라 각 사용자가 언제 규칙을 만나는지, 즉 제품의 가치를 느끼기 전인지 후인지를 묻게 됩니다.

코드를 통해 제품이 무엇을 하는지는 알았습니다.

- 하지만 그 규칙이 좋은지, 실제로 계정을 보호하는지, 사용자가 치르는 비용을 받아들일 만한지는 코드가 말해 주지 않습니다.

결정하려면 컨텍스트의 다른 요소가 여전히 필요합니다.

- 연구에서 관찰한 내용과 그 범위,
- 보안 목표,
- 이 규칙을 선택하게 된 이유,
- 변경할 때 예상되는 결과입니다.

스킬은 제품에 관한 사실 하나를 확립합니다. 컨텍스트는 그 사실을 기존 노트와 결정에 연결하는 동시에, 기원을 확인할 수 있는 출처를 보존합니다. 판단의 책임은 여전히 제게 있습니다.

비밀번호 생성 방식을 바꾸기로 결정하면, 결정 기록이 선택과 그 범위, 근거가 된 이유를 보존합니다. 구현 뒤에는 스킬이 제품으로 돌아가 새로운 동작이 결정과 일치하는지 확인할 수 있습니다.

따라서 움직임은 양방향입니다. 제품은 사고를 밝히고, 결정은 다시 제품을 바꿉니다.

```
관찰 —> 아이디어 —> 결정
  ▲                  |
  |                  ▼
검증된 제품 <——— 변경된 제품
```

왜를 보존하고 무엇을 재구성하기

코드 중심 접근법

코드는 더 이상 개발의 최종 결과물에 그치지 않습니다. 제품의 현재 동작을 설명하는 내용을 재구성하는 출처가 됩니다. 컨텍스트는 코드만으로 설명할 수 없는 것, 즉 출처와 추론, 결정, 배제한 대안, 선택한 이유를 보존합니다.

이런 분리는 문서의 자리를 바꿉니다.

제품이 왜 이 방향으로 갔는지 설명하는 내용은 보존합니다. 인터뷰와 제약, 가설, 판단이 여기에 해당합니다. 제품의 현재 동작을 설명하는 내용은 제품에서 다시 구성합니다.

- 고객 지원팀에 보낼 답변,
- 기능 설명 문서,
- 변경 기록,
- 그 밖의 결과물입니다.

같은 동작을 설명하는 두 버전, 즉 코드에 있는 버전과 공개되는 순간부터, 사실은 그보다 훨씬 전부터 낡기 시작하는 별도 문서의 버전을 더 이상 수작업으로 함께 관리하지 않습니다.

그렇다고 모든 것이 코드에서 나오지는 않습니다.

- 사용자 연구는 코드 안에 없습니다.
- 규제 의무는 고유한 출처를 유지합니다.
- 결정은 그 결과로 생긴 동작만으로 설명되지 않습니다.

코드는 구현된 무엇을 담습니다. **컨텍스트**는 왜를 보존하고, 상황이 바뀌면 다시 논의할 수 있게 합니다.

6개월 뒤 돌아오기

고객 지원팀에 전달한 답은 오늘 조사한 제품 버전에 연결되어 있습니다. 이후 버전까지 자동으로 설명하지는 않습니다.

- 6개월 뒤에는 초대 사용자 여정이 달라졌을 수 있습니다.
- 기능 플래그가 다른 변형을 활성화했거나 새로운 로그인 방식이 생겼을 수도 있습니다.

컨텍스트는 질문과 기능 관점의 답, 확인 날짜, 살펴본 버전, 인용한 근거, 명시한 한계를 보존합니다. 따라서 당시 우리가 추론할 때 알고 있던 내용을 다시 찾을 수 있습니다.

현재 동작을 알고 싶으면 스킬에 검증을 다시 실행해 달라고 요청하면 됩니다.

- 예전 답을 복사하지 않고 실제 제품에서 다시 시작합니다.

그러면 흔히 뒤섞이는 두 질문을 구분할 수 있습니다.

- 당시 우리는 왜 이 결정을 내렸는가?
- 제품은 오늘 무엇을 하는가?

컨텍스트는 추론을 보존해 첫 번째 질문에 답합니다. 스킬은 사실의 기준이 되는 원천으로 돌아가 두 번째 질문에 답합니다.

따라서 제품은 컨텍스트를 대체하지 않으며, 컨텍스트도 제품을 담고 있다고 주장하지 않습니다.

- 한쪽은 확인해야 할 현실을 제공합니다.
- 다른 쪽은 그 현실에 의미를 부여하고 발전시키는 데 필요한 지식을 보존합니다.

이러한 순환은 외부 도구에서 끝나지 않습니다. 여러 컨텍스트가 같은 제품의 서로 다른 측면을 다룰 때는, 하나의 폴더에 합치지 않고도 각자의 지식을 서로 찾을 수 있어야 합니다.

기억할 점 – 코드 중심 접근법

제품을 지속적으로 설명하기 위해 별도의 명세를 관리하지 않습니다. 제품의 동작을 설명하는 내용은 코드에서 다시 구성합니다. 답의 가치는 함께 제시되는 근거와 명시된 한계에서 나옵니다. 코드는 구현된 무엇을 담고, 컨텍스트는 왜를 보존해 다시 논의할 수 있게 합니다.

6.3

컨텍스트가 세 개가 된 날

한 현장 기술자가 작업 보고서에 빠진 정보가 있어서 고객에게 다시 전화해야 했다고 말합니다.

저는 그 말을 기록합니다. 그러고는 멈춥니다. 어디에 넣어야 할지 모르기 때문입니다.

- 작업 보고서의 품질에 관한 컨텍스트가 있습니다. 이 책 후반부 내내 다룬 바로 그 컨텍스트입니다.
- 현장 작업 자체에 관한 두 번째 컨텍스트도 있습니다. 현장 작업에 무엇이 포함되는지, 어떤 규칙이 적용되는지, 이미 무엇을 결정했는지를 담습니다.
- 몇 주 전에 연 더 넓은 세 번째 컨텍스트도 있습니다. 작업자가 1차 대응을 맡도록 어떻게 이끌 것인가를 다룹니다.

현장 기술자의 말은 세 컨텍스트 모두와 관련됩니다.

- 문제가 있는 작업 보고서를 기록하고,
- 현장 작업 자체에 관한 내용을 말하며,
- 작업자에 관한 질문을 밝힙니다.

이 순간부터 문제의 성격이 달라집니다.

- 컨텍스트가 하나뿐일 때 질문은 “이 내용이 들어가는가, 아니면 주제에서 벗어나는가?”였습니다.
- 세 개가 되자 질문은 “어느 곳에 넣어야 하는가?”로 바뀝니다.

잘못될 수 있는 두 가지 방식

첫 번째 유혹은 모든 것을 한데 모으는 것입니다.

- 거대한 공간 하나에 모든 지식을 넣으면 어디에 정리할지 고민하지 않아도 됩니다.

하지만 제가 피하려 했던 것이 바로 그것입니다. 모든 것을 담은 하나의 공간은 문서 시스템이 늘 그랬던 모습으로 되돌아갑니다. 그곳에서는

- 정보는 존재하지만 열 때마다 다시 선별해야 하고,
- 목표는 뒤섞이며,
- 출처는 계속 쌓이고,
- 어느 노트도 분명한 주인을 갖지 못합니다.

책의 앞부분에서 이미 이 막다른 길을 이야기했습니다. 새로운 용어로 다시 만든다고 나아지지는 않습니다.

두 번째 유혹은 정반대입니다. 각 컨텍스트가 자기 영역 안에만 머무르게 하는 것입니다. 각각은 깔끔하고, 경계가 분명하고, 일관되지만 서로를 보지 못합니다.

결과는 금세 드러납니다. 작업 보고서 컨텍스트가 세 세션에 걸쳐 하나의 규칙을 안정된 형태로 정리했는데, 현장 작업 컨텍스트에는 이미 같은 규칙이 있었습니다. 눈에 보이는 중복 파일은 아니었습니다. 아무도 알아차릴 수 없는 채로 같은 일을 다시 한 것입니다.

따라서 세 번째 길이 필요합니다. 그 길은 각 컨텍스트가 무엇을 담는지 아는 데서 시작합니다.

이름 앞의 숫자

제 컨텍스트 이름은 `01-saisie-compte-rendu`(작업 보고서 입력), `02-intervention`(현장 작업), `03-autonomie-operateurs`(작업자 자율성), `04-exploitation-catch-all`(운영 캐치올)입니다.

숫자는 순위가 아니라 이름을 붙이는 규칙입니다. 컨텍스트를 열기 전부터 무엇을 담는지 알려 줍니다.

- **`02-`는 제품의 엔터티입니다.** 운영 환경에서 실행되며 고유한 규칙을 가진 소프트웨어의 일부입니다. ERP의 `02-facturation`(청구), 제 제품의 `02-intervention`이 그 예입니다. 그 안의 내용은 확립된 사실입니다. 이 컨텍스트는 닫히지 않습니다.

- **01-**은 엔터티에 관한 가설입니다. 01-saisie-compte-rendu는 입력 방식이 바뀌어야 한다고 생각하지만 아직 어떻게 바꿀지는 모른다는 뜻입니다. 그 안의 어떤 내용도 확립된 사실이 아닙니다. 그래서 운영 중인 안정된 내용과 가설을 명확히 구분하려고 01- 컨텍스트를 만듭니다.
- **03-**은 문제입니다. 03-autonomie-operateurs는 작업자가 1차 대응을 맡게 하는 문제입니다. 기능 하나가 아니라 여러 엔터티에 걸쳐 있으며, 여기서 여러 01-이 생겨납니다.
- **04-**는 캐치올입니다. 우연히 들은 말, PDF, 직관처럼 아직 어떤 컨텍스트에도 주제가 정해지지 않은 재료가 들어갑니다. 이것이 없으면 아이디어 조각 하나마다 컨텍스트를 만들어야 합니다. 저도 처음에는 그렇게 했다가 몇 주 만에 내용 없는 컨텍스트가 50개 넘게 생겼습니다.

01-은 언제나 활성 작업에서 빠져나간다

새로운 입력 기능이 운영 환경에 배포되는 날, 더는 가설이 아니며 작업의 일부가 됩니다.

이제 제품을 설명하는 내용은 02-intervention에 반영하도록 제안됩니다. 나머지는 탐색용 컨텍스트에 남고, 그 컨텍스트는 보관됩니다. 작업 보고서가 어떻게 동작하는지 찾을 활성 공간이 두 곳으로 나뉘지 않습니다.

결과로 이어지지 않은 가설은 무엇 때문에 중단됐는지와 함께 보관됩니다. 지워야 할 실패가 아니라, 2년 뒤 같은 길을 다시 걷지 않게 해 줄 기록입니다.

04-는 반대 방향으로 비워집니다. 어떤 문제가 분명해지는 날 그 문제에 맞는 컨텍스트를 만든 뒤, 캐치올이 그 문제와 관련해 무엇을 보관하고 있었는지 다시 살펴봅니다. 쓸모없는 내용은 삭제합니다.

이 모든 방식은 제가 조사 중인 것과 확립된 것을 구분하여 “이것을 어디에 넣을까?”라는 질문 하나를 해결합니다.

- 이 구분이 없으면 포기한 가설마다 제품 설명에 흔적을 남깁니다.
- 그리고 모든 내용이 참이어야 하는 곳에서는 감히 어떤 가설도 세우지 못하게 됩니다.

알아차리는 데 몇 달이 걸린 오류

처음에는 `03-tarification`(가격 책정)을 열었습니다. 당연해 보였습니다. 큰 주제이고, 여러 엔터티와 관련되며, 오래 이어지기 때문입니다.

6개월 뒤 그 안에는 다음 내용이 있었습니다.

- 가격표,
- 경쟁사 벤치마크 두 건,
- 할인에 관한 논의,
- 잘못된 청구서에 관한 고객 지원 티켓,
- 벨기에 부가가치세에 관한 노트,
- 경쟁사 화면 캡처 세 장,
- 그 밖의 자료입니다.

어느 것도 주제 밖에 있지 않았습니다. 모두 가격 책정에 관한 내용이었습니다.

바로 그것이 문제였습니다. 자료 하나를 거절하려면 "이것은 나에게 도움이 되지 않는다"고 말할 수 있어야 합니다. 하지만 이 컨텍스트는 아무것도 찾고 있지 않았습니다. 질문 없이 이름표만 있었습니다.

반면 `03-autonomie-operateurs`에는 질문이 있습니다. 작업자가 1차 대응을 맡도록 어떻게 이끌 것인가입니다.

따라서 `03-`은 큰 주제가 아니라 질문입니다. 그렇다고 가격 책정이 사라진 것은 아닙니다. `02-facturation`이 규칙을 담고, 하나의 `01-`이 그 규칙에 의존할 수 있듯이, 가격 책정은 어느 하나를 지배하지 않은 채 여러 컨텍스트에 걸쳐 있습니다.

그러므로 제 규칙에는 업무 영역을 위한 칸도, 그 밖의 많은 것을 위한 칸도 없습니다. 명백한 자리가 있는 것만 정리하는 데 쓰이며, 그 이상은 아닙니다. 나머지는 캐치올에서 기다립니다. 어디에 넣었는지 기억하는 대신 무엇을 찾는지 설명해서 다시 찾습니다.

실수를 통해 배운 결과 캐치올도 두 개가 됐습니다. 하나만 두었을 때는 제품 업무에서 나온 내용과 산업 운영 영역에서 나온 내용이 뒤섞였습니다. 두 영역은 용어도, 독자도, 같은 대상을 부르는 방식도 달랐습니다. 비워지는 속도보다 읽기 어려워지는 속도가 더 빨랐습니다.

- `04-exploitation-catch-all`에는 고객의 업무 영역, 즉 운영에 관한 기능적 내용이 들어갑니다.
- `04-productops-catch-all`에는 제 직무와 그 조직 방식, 즉 Product Ops에 관한 내용이 들어갑니다.

각각 읽을 수 있는 상태를 유지하므로, 각각 비워질 수 있습니다.

현장 기술자의 말은 캐치올을 거칠 필요가 없었습니다. 지금 제가 던지는 질문과 상관없이 제품에 관해 참인 내용이 존재하는 곳, 즉 `02-intervention`으로 들어갔습니다.

이제 남은 것은 제 세 컨텍스트가 그 사실을 어떻게 알아차리는가입니다.

기억할 점 – 네 가지 접두사와 하나의 질문

이름 앞의 숫자는 컨텍스트가 무엇을 담는지 알려 줍니다. `02-`는 실행 중인 엔터티, `01-`은 그 엔터티에 관한 가설, `03-`은 여러 엔터티에 걸친 문제, `04-`는 아직 어디에 둘지 모르는 내용을 넣어 두는 임시 보관소입니다. `01-`은 이제 제품을 설명하는 내용을 자신의 `02-`에 반영하도록 제안한 뒤 보관되기 위해 존재합니다. 결과로 이어지지 않았다면 중단된 이유와 함께 보관됩니다. 업무 영역은 컨텍스트가 아닙니다. 여러 컨텍스트에 걸쳐 있지만 어느 하나도 지배하지 않습니다. 질문이 없는 컨텍스트는 무엇이 주제 밖인지 영원히 판단할 수 없습니다.

6.4

이웃집 자료를 읽되 그 집으로 이사 가지 않기

작업 보고서의 품질을 다루는 컨텍스트인 01-saisie-compte-rendu의 작업을 다시 시작합니다. 계속하려면 현장 작업 자체의 규칙을 알아야 합니다. 현장 기술자가 어떤 내용을 입력할 수 있는지, 무엇이 필수인지, 무엇이 이미 결정됐는지를 알아야 합니다.

이 규칙은 제 컨텍스트 안에 있지 않습니다. 실제 제품에 존재하는 내용을 설명하는 이웃 컨텍스트, 02-intervention에 안정된 형태로 보존되어 있습니다.

곁에 두고 쓰려고 복사할 수도 있습니다. 그러나 6개월 뒤에는 같은 규칙의 두 가지 버전이 생기고 어느 쪽을 믿어야 하는지 쉽게 알 수 없을 것입니다.

그래서 더 가벼운 동작부터 시작합니다. 제 컨텍스트가 02-intervention을 읽는다고 선언합니다.

읽기는 지식을 소유자에게 남겨 둔다

이 관계는 임무와 지침 옆, 제 컨텍스트의 안정된 프레임에 기록됩니다. 이를 통해 02-intervention이 이미 알고 있는 내용을 살펴볼 수 있는 명확한 권한을 얻습니다.

재료는 소유자에게 남습니다. 그곳에서 계속 변하며, 제 탐색은 이를 소유한다고 주장하지 않고 활용합니다.

따라서 작업 보고서에 관한 모든 연구에 규칙을 복제하지 않고도 필수 입력란의 현재 규칙을 확인할 수 있습니다. 그 규칙을 만든 출처와 결정으로 돌아갈 수도 있습니다.

읽기 권한은 수정 권한이 아닙니다. 노트가 불완전하거나 틀렸다고 생각되면 제 컨텍스트가 제안을 작성할 수는 있지만, 이웃의 내용을 직접 고치지는 않습니다.

이 첫 번째 관계는 외부 지식이 담당자의 영역에서 계속 살아가도록 두면서 활용해야 하는 흔한 필요에 답합니다.

하지만 모든 경우를 해결하지는 못합니다.

흡수는 지식을 독립시킨다

때로는 읽는 것만으로 충분하지 않습니다.

결정의 근거가 된 정확한 규칙을 고정해야 할 수 있습니다. 제 탐색의 특정 범위에 맞게 다시 표현하고 싶을 수도 있습니다. 원래 컨텍스트가 보관되거나 이동하거나 재구성되더라도 그 지식을 계속 쓸 수 있어야 할 때도 있습니다.

그럴 때는 흡수를 선택합니다.

흡수하면 지식이 제 컨텍스트 안으로 들어옵니다. 노트를 그대로 가져오거나 제 용어로 다시 표현할 수 있습니다. 필요에 따라 검증에 필요한 출처도 복사하거나 그 출처를 가리키는 참조를 보존합니다.

복사본이라고 해서 과거 없는 아이디어가 되지는 않습니다. 어느 컨텍스트에서 왔는지, 언제 흡수했는지, 무엇을 다시 표현했는지를 보존합니다.

따라서 차이는 기술적 동작보다 그 결과 생기는 책임에 있습니다.

- 읽을 때 지식은 계속 이웃에게 속합니다.
- 흡수할 때는 독립된 버전이 제 영역으로 들어오며, 제가 그 사용에 책임을 집니다.

읽기는 의존 관계를 만듭니다. 흡수는 새로운 책임을 만듭니다.

흐름은 반대 방향으로도 갈 수 있다

제 탐색은 `02-intervention`이 이미 아는 내용을 가져다 쓰기만 하지 않습니다. 탐색을 통해 관찰과 가설, 때로는 오래가는 지식도 만들어집니다.

작업 보고서의 품질을 연구하다가 어떤 입력 규칙이 사실 현장 작업 자체에 속한다는 것을 발견할 수 있습니다. 이 지식이 나중에 보관될 수도 있는 탐색용 컨텍스트 안에 갇혀서는 안 됩니다.

그렇다고 제가 `02-intervention`에 직접 쓰지는 않습니다. 그곳에 제안합니다.

흐름의 방향이 바뀌었습니다.

```
01-saisie-compte-rendu ── 읽기 ──▶ 02-intervention

01-saisie-compte-rendu ── 제안 ──▶ 02-intervention
                                        │
                                        └── 흡수 여부 결정
```

지식을 만든 쪽이 그 지식의 최종 위치까지 결정하지는 않습니다. 받는 컨텍스트가 자신의 임무에 따라 평가합니다.

다음과 같이 처리할 수 있습니다.

- 노트를 그대로 가져오고,
- 자신의 용어로 다시 표현하며,
- 두 해석이 모두 유효하면 차이를 보존하고,
- 탐색에는 유용하지만 현장 작업 자체를 설명하지 않는다면 흡수를 거절합니다.

거절은 지식 순환의 실패가 아닙니다. 이 컨텍스트에 한정된 결론이 아무도 모르게 일반적인 진실이 되는 것을 막는 장치입니다.

탐색이 제품의 현실이 될 때

몇 주 뒤 새로운 작업 보고서 입력 방식이 운영 환경에 배포됩니다. `01-saisie-compte-rendu`는 자신의 역할을 다했습니다. 가설에 불과했던 것이 제품의 현실이 됐습니다.

그렇다고 그 안의 내용을 전부 `02-intervention`으로 옮기지는 않습니다. 탐색이 만들어 낸 내용을 검토해 두 집합으로 나눕니다.

첫 번째 집합은 이제 제품을 설명하거나 그 이유를 밝히는 내용입니다.

- 새로운 입력 규칙을 확립하는 노트,
- 그 규칙을 부르는 안정된 용어,

- 구조를 정한 결정과 그 선택의 이유,
- 배포 뒤에도 열린 채로 남은 모순,
- 이 지식을 검증하는 데 필요한 출처 또는 참조입니다.

이 요소들은 `02-intervention`에 제안됩니다. 안정된 컨텍스트는 이를 맹목적으로 쌓지 않습니다. 기존 노트를 더 풍부하게 하고, 필요한 내용은 다시 표현하며, 아이디어가 실제로 새로울 때만 새 노트를 만듭니다.

이 사례에서는 작업 완료에 필요한 정보를 네 가지로 제한하는 규칙이 그 이유를 설명하는 결정과 함께 `02-intervention`으로 들어갑니다. 이제 제품의 현실을 설명합니다.

두 번째 집합은 현재 제품이 아니라 개발 과정을 이야기합니다.

- 포기한 가설,
- 제외한 모형과 방향,
- 중간 초안,
- 탐색에만 해당하는 작업,
- 지나온 길을 이해하게 해 주는 논의입니다.

이 요소들은 `01-saisie-compte-rendu`에 남습니다. 쓸모가 없어진 것이 아닙니다. 무엇을 시도했고, 왜 어떤 방향을 거절했으며, 새로운 이유 없이 무엇을 다시 해서는 안 되는지 설명합니다.

선별이 끝나면 탐색용 컨텍스트 전체를 보관합니다. 계속 읽을 수 있고 나중에 작업을 다시 이어 갈 수도 있지만, 더는 활성 작업을 설명한다고 주장하지 않습니다.

탐색이 결과로 이어지지 않았더라도 포기한 이유와 함께 보관됐을 것이며, `02-intervention`에는 아무것도 강요하지 않았을 것입니다.

따라서 운영 환경에 배포한다고 두 컨텍스트가 합쳐지는 것은 아닙니다. 검토가 시작됩니다. 이제 제품을 설명하는 내용은 안정된 컨텍스트로 들어가고, 탐색 과정을 이야기하는 내용은 보관된 컨텍스트에 남습니다.

이 선별은 양쪽을 모두 보호합니다.

- 탐색은 자유롭게 틀릴 수 있습니다.
- 안정된 컨텍스트는 자신의 것으로 인정하기로 선택한 내용만 받습니다.

관계도 프레임의 일부다

읽기와 흡수, 제안을 암묵적인 동작으로 남겨 두어서는 안 됩니다.

컨텍스트의 안정된 프레임은 다음 내용을 밝힙니다.

- 어떤 다른 컨텍스트를 살펴볼 수 있는지,
- 그 열람으로 무엇을 할 수 있는지,
- 어떤 규칙에 따라 지식을 흡수할 수 있는지,
- 그 기원과 다시 표현한 내용을 어떻게 보존하는지입니다.

이런 거버넌스에 무거운 기반 시설이 필요한 것은 아닙니다. 기원과 책임, 경계가 계속 보이도록 관계에 자리를 마련할 뿐입니다.

따라서 네트워크는 모든 것을 서로 연결해서 만들어지지 않습니다. 작업에 실제로 이웃이 필요할 때마다 관계를 하나씩 맺으며 만들어집니다.

- 이웃의 자료를 읽으면 그곳의 지식을 복제하지 않을 수 있습니다.
- 흡수하면 제 영역에서 살아야 하는 지식을 독립시킬 수 있습니다.
- 제안하면 이웃이 제 탐색에서 나온 내용을 받아들일지 결정할 수 있습니다.

종이 위에서 이런 순환은 깔끔해 보입니다. 그러나 대가가 있습니다. 참조는 아무런 경고 없이 낡을 수 있고, 복사본은 출처 정보를 잃을 수 있으며, 두 컨텍스트는 서로 다른 해석을 보존할 수 있습니다.

다음 장에서는 이 문제를 살펴보겠습니다.

기억할 점 – 세 가지 동작, 세 가지 책임

읽기는 소유자에게 남아 있는 지식을 살펴봅니다. 흡수는 받는 컨텍스트 안에 출처가 달린 독립된 버전을 만듭니다. 탐색은 배운 것을 제안하지만, 어떤 내용을 받아들일지는 대상 컨텍스트의 소유자가 결정합니다. 운영 환경 배포는 폴더를 합치지 않습니다. 안정된 컨텍스트는 이제 제품을 설명하는 내용을 흡수하고, 탐색은 나머지를 보존한 뒤 보관됩니다.

6.5

지식 순환의 대가

6주가 지났습니다.

작업 보고서에 관한 제 작업을 검토에 넘길 준비가 됐는데, 누군가 차이를 발견합니다. 제가 의지한 규칙, 즉 작업 완료 시 어떤 입력란이 필수인지 정한 규칙이 더는 현재 제품의 규칙과 같지 않습니다.

규칙이 바뀌었습니다. 제가 작업하는 동안 다른 탐색이 결과로 이어졌고, 현장 작업 컨텍스트가 그 결과를 흡수했습니다.

아무도 제게 알려 주지 않았습니다. 6주 전에 이미 읽었고 다시 확인해야 한다는 신호도 없었기에 다시 읽지 않았습니다.

앞 장에서는 깔끔해 보이는 규칙을 세웠습니다. 읽을 때는 참조하고, 흡수할 때는 복사하는 규칙입니다. 저는 이를 이미 끝난 판단으로 여겼습니다.

그러나 위험이 사라진 것은 아닙니다. 위험의 위치가 옮겨졌을 뿐이며, 어느 쪽에 생기는지 아는 편이 낫습니다.

눈에 보이지 않는 의존 관계

방금 일어난 일에는 이름이 있습니다. 보이지 않는 의존 관계입니다.

연결이 살아 있으므로 제가 읽는 내용은 저도 모르게 바뀔 수 있습니다. 그 자체는 문제가 아닙니다. 오히려 제가 원하는 모습일 때가 많습니다. 규칙이 소유자의 영역에서 변화한다면 시스템이 제대로 작동하는 것입니다.

문제는 규칙이 바뀌었다는 사실을 모르는 데 있습니다.

무엇이 부족한지는 간단합니다. 제가 읽는 내용을 아무도 감시하지 않습니다. 제 컨텍스트는 누구에게 의존하는지는 알지만, 의존하는 대상이 언제 바뀌었는지는 알지 못합니다. 작동 방식의 절반만 있는 셈이며, 빠진 절반에서 비용이 생깁니다.

복사했다면 이 문제에서는 보호받았을 것입니다. 받는 컨텍스트는 독립하며, 원본의 존재나 변화에 더는 의존하지 않습니다.

대신 다른 것을 잃습니다. 복사본의 출처가 표시되지 않으면 기원이 사라집니다. 1년 뒤에는 옳은 노트가 있어도 무엇에 근거했는지 알 수 없습니다. 바로 이 책 전체의 추적성이 막으려는 상태입니다.

따라서 어느 한쪽을 고르는 것이 결론은 아닙니다. 각 동작이 무엇을 수반해야 하는지 기록해야 합니다.

- 참조에는 원본이 언제 바뀌는지 아는 장치가 필요합니다.
- 복사에는 어디서 왔는지, 즉 어느 컨텍스트에서 언제 가져왔고 그 과정에서 무엇을 다시 표현했는지 밝혀야 합니다.

발전 가능성

앞으로 나올 또 다른 답은 기술적인 방법일 것입니다. 컨텍스트가 상태의 변화, 특히 다른 컨텍스트가 관찰하는 범위의 변화를 관찰자에게 알리는 기능입니다. 컨텍스트는 아직 탐색할 가능성이 많은 살아 있는 주제입니다.

읽기로 얻지 못하는 권한

6

어떤 컨텍스트를 자주 읽다 보면 마치 조금은 소유한 것처럼 느껴집니다. 이웃의 노트가 불완전하고 무엇을 넣어야 할지도 알며 30초면 고칠 수 있어서, 직접 수정하고 싶었던 적이 여러 번 있습니다.

그래도 규칙은 달라지지 않습니다. 제가 전할 내용은 제안이며, 결정은 받는 쪽의 몫입니다.

추적성을 깨뜨리지 않고 삭제하기

삭제에는 별도의 규칙이 필요하며, 저는 보관함을 비우다가 이를 배웠습니다.

노트가 다른 컨텍스트로 이동해도 그 노트를 만든 출처는 뒤에 남습니다. 그러면 더는 어떤 노트도 연결되지 않은 채 홀로 남을 수 있습니다.

정리하고 싶어지는 것이 자연스럽습니다. 절반만 옳은 생각입니다.

이미 읽고 노트로 변환한 출처는 삭제할 수 있습니다. 그 내용이 다른 곳에서 살아 있기 때문입니다. 아직 아무도 읽지 않은 출처는 삭제할 수 없습니다.

- 누구도 추출하지 않은 재료를 담고 있으며, 버리는 것은 한 번도 열어 보지 않은 내용을 잃는 일입니다.

따라서 기준은 파일의 나이도, 차지하는 공간도 아닙니다.

- 읽은 상태와 아직 그 출처를 인용하는 항목을 함께 살펴봐야 합니다.

여기에는 타협할 수 없는 조건이 하나 따릅니다. 삭제하기 전에 어떤 노트나 용어 항목, 초안, 이력의 어느 줄에서도 그 출처를 더는 인용하지 않는지 확인해야 합니다. 확인하지 않으면 추적성을 지키겠다던 규칙이 오히려 추적성을 깨뜨립니다.

차이가 곧 오류는 아니다

두 컨텍스트가 같은 주제에 관해 서로 다른 내용을 말하는 경우가 남습니다.

이를 불일치로 보고 빨리 어느 한쪽을 골라 질서를 되찾고 싶어지는 것이 자연스럽습니다. 하지만 흔히 잘못된 판단입니다.

범위가 달라서 생긴 차이일 수 있습니다. 각자의 영역에서는 둘 다 옳습니다. 한쪽만 조사하고 다른 쪽은 알지 못하는 열린 가설일 수도 있습니다. 서로 다른 위치에서 주제를 바라보기 때문에 두 해석이 모두 성립할 수도 있습니다.

그렇다면 차이를 그대로 보존할 가치가 있으며, 그렇게 할지는 사람이 결정합니다. 시스템이 보장해야 하는 것은 그 차이가 존재한다는 사실을 우리가 아는 것입니다.

잊힌 차이는 언제나 가장 나쁜 순간에 모습을 드러냅니다. 다른 버전이 존재하는지 모르는 사람이 둘 중 하나를 근거로 삼을 때입니다. 작업 완료 규칙에서 제게 일어난 일이 바로 그것입니다. 다만 두 가지 해석이 충돌한 것이 아니라 하나의 해석이 낡은 경우였습니다.

따라서 네트워크를 유지하려면 대가를 치러야 하며, 검토 단계에서 발견하는 것보다 미리 이름 붙이는 편이 낫습니다. 두 컨텍스트를 연결한다고 주의 깊게 살피는 일이 사라지지는 않습니다. 그 일을 사람에게 옮길 뿐입니다.

결국 이 책이 처음부터 이야기해 온 내용이기도 합니다.

기억할 점 – 네트워크의 대가

참조는 아무런 경고 없이 낡을 수 있고, 복사본은 기원을 잃을 수 있습니다. 모든 동작에는 부채가 있으며 어떤 부채인지 기록해야 합니다. 읽기는 쓰기를 허용하지 않습니다. 이웃에게 전할 내용은 언제나 제안이지 직접적인 수정이 아닙니다. 이미 노트로 변환한 출처는 사라질 수 있지만, 한 번도 읽지 않은 출처는 절대 삭제해서는 안 됩니다. 무엇이 아직 인용하는지 확인하지 않고는 아무것도 삭제하지 않습니다. 두 컨텍스트가 서로 모순된다고 반드시 어느 쪽이 틀린 것은 아닙니다. 심각한 문제는 그 사실을 모르는 것입니다.

책을 덮기 전에

맺음말

맺음말

책은 여기서 끝나지만, 이 주제는 책과 함께 끝나지 않습니다.

에필로그에서는 이 작업의 현재 상태와 아직 겉만 훑은 부분, 앞으로의 내용이 공개될 곳을 이야기합니다. 이어지는 글에서는 컨텍스트에 앞서 나왔거나 그 주변에 있는 접근법 속에서 이 시스템의 위치를 살펴봅니다. 그로써 이 시스템이 무엇을 빌려왔고 무엇을 받아들이지 않았는지 알 수 있습니다.

7.1

에필로그

여기까지 와 주셔서 감사합니다.

이 책을 가능한 한 읽기 쉽게 만들려고 했지만, 완전히 성공하지는 못했습니다. 제가 원했던 것보다 딱딱한 대목도 남아 있고, 이야기보다 작동 방식이 앞서는 곳도 있습니다. 그래도 끝까지 견뎌 주셨습니다. 그 성의를 가볍게 여기지 않습니다.

이 책은 주제의 겉면만 훑었다

이렇게 두꺼운데도 컨텍스트를 피상적으로만 다룬 책입니다.

몇몇 부분은 개념적으로 충분히 깊지 않으며, 저는 이 책을 내보내면서 그 점을 알고 있습니다. 원자 노트가 한 예입니다. 구현 방식에 따라 결과의 편차가 매우 큽니다. 컨텍스트 안에서 작동하는 방식은, 자료의 양과 용도가 전혀 다른 LLM 위키로 넘어가면 수정이 필요합니다.

이런 부분은 더 단단해져야 합니다. 여기서 그렇게 하지 못한 것은 주제가 끝나서가 아니라, 책은 언젠가 멈춰야 하기 때문입니다.

곁에 두고 계속할 프로젝트

컨텍스트는 계속 발전하겠지만, 앞으로도 사이드 프로젝트로 남을 것입니다.

이는 본업 가장자리에서 하는 활동이고, 저는 이미 너무 많은 저녁을 이 일에 썼습니다. 다시 돌아올 겁니다. 언제나 그랬듯이 다시 돌아올 겁니다. 하지만 이제는 속도를 낮추고 삶을 즐길 때입니다.

다음에 탐구하고 싶은 것

선행 연구 조사는 했습니다. 제 개념과 매우 비슷한 개념을 찾았습니다. 때로는 제품과 지식 관리에서 멀리 떨어진 분야에서 나왔고, 종종 다른 언어로 작성된 연구였습니다. 특히 일본어와 중국어로 쓴 연구는 프랑스어권

과 영어권에서 대부분 보이지 않습니다. 이 조사 과정에 관한 글을 써 두었고, 블로그에 공개할 예정입니다.

아직 부족한 것은 다음 단계, 즉 이 연구들을 분석하는 일입니다.

이름도 알고 논문도 있지만, 저는 아직 그것을 읽지 않았습니다. 수백 명이 수십 년에 걸쳐 연구한 결과입니다. 저는 이 문제를 혼자 저녁 시간에 다뤘습니다. 그 연구를 찬찬히 검토하는 편이 이 틀을 계속 혼자 발명하는 것보다 훨씬 확실하게 강화해 줄 것입니다.

다음은 후속편이 아닐 것이다

이 책은 개념을 구체적인 형태로 굳히는 데 쓰였습니다. 흩어져 있던 것을 모아 다른 사람에게 전할 수 있게 하고, 함께 논의할 수 있을 만큼 안정적인 형태를 부여하는 것이 역할이었습니다.

당분간 발전 내용은 후속편이나 이 책의 새 판본이 아니라 `malorean.net`에 공개할 것입니다. 글은 고치고, 보충하고, 앞선 글을 반박할 수 있습니다. 책은 내용을 고정합니다. 여기서 설명한 내용은 두 번 고정하기에는 아직 충분히 성숙하지 않았습니다.

이후의 여정을 따라오고 싶다면 그곳에서 계속 쓰겠습니다.

7.2

계보와 이웃한 접근법

컨텍스트는 지적인 진공 속에서 나타나지 않았습니다.

컨텍스트를 이루는 작동 방식 대부분은 지식 관리, 노트 작성법, 소프트웨어 공학, 의사 결정 지원 시스템, 인지과학, 조직 기억 등 여러 분야에 뿌리를 두고 있으며, 그중에는 아주 오래된 것도 있습니다.

이 가운데 일부는 처음부터 의식한 영향이었습니다. 다른 일부는 방법론을 설계한 뒤, 책이 거의 완성됐을 때 선행 연구를 조사하는 과정에서야 발견했습니다.

따라서 두 가지를 구분해야 합니다.

영향은 제가 이미 알고 있었고 그 원칙 일부를 의식적으로 가져오거나 변형한 방법 또는 아이디어입니다.

계보적 유사성이나 인접 관계는 컨텍스트에 직접 영향을 주지는 않았지만, 때로는 놀랄 만큼 비슷한 작동 방식으로 유사한 문제를 해결한 앞선 시스템을 뜻합니다.

이 구분은 중요합니다. 두 아이디어가 서로 가까워도 하나가 다른 하나에서 나온 것은 아닐 수 있습니다.

제텔카스텐: 재사용할 수 있는 단위의 네트워크로서의 지식

제텔카스텐과의 유사성이 아마 가장 뚜렷할 것입니다.

니클라스 루만의 시스템에서 노트는 그때 진행 중인 프로젝트의 폴더 안에 단순히 정리되지 않습니다. 서로 연결된 노트의 지속적인 시스템으로 들어가 새로운 추론에 다시 쓰일 수 있습니다.

컨텍스트도 이 생각을 강하게 공유합니다. 작업하면서 만들어진 지식이 그 결과물과 함께 사라져서는 안 됩니다.

컨텍스트의 원자 노트는 분명 이 계보에 속합니다.

가장 큰 차이는 노트의 지위에 있습니다. 원자 노트는 출처를 보존해야 하며, 출처와 모순, 결정, 현재 상태, 종합, 결과물을 명시적으로 구분하는 구조 안에 놓입니다.

따라서 제텔카스텐은 지식 자산을 점진적으로 구축하는 방식과 매우 가까운 계보를 제공하지만, 시스템 전체를 설명하지는 않습니다.

PARA: 행동에 맞춰 정보 정리하기

티아고 포르테의 PARA는 컨텍스트에 의식적으로 영향을 준 방법 가운데 하나입니다.

PARA는 정보를 Projects, Areas, Resources, Archives라는 네 범주로 정리합니다. 그 중요한 통찰 가운데 하나는 정보 구조가 세계를 추상적으로 분류하는 대신 행동을 지원해야 한다는 것입니다.

컨텍스트도 이 통찰의 일부를 이어받았습니다.

어떤 주제가 존재한다는 이유만으로 컨텍스트가 존재하지는 않습니다. 사람이 하나의 임무를 추구하기 때문에 존재합니다.

따라서 경계는 "데이터베이스에 관해 내가 아는 모든 것"이 아닙니다.

"제품 X에 사용할 데이터 아키텍처 선택하기"가 될 수 있습니다.

차이는 컨텍스트 자체가 고유한 상태와 출처, 지식, 결정, 생애 주기를 갖춘 독립적인 작업 단위가 된다는 데 있습니다.

도메인 주도 설계: 내용보다 먼저 경계를 세우기

컨텍스트라는 이름은 자연스럽게 도메인 주도 설계의 바운디드 컨텍스트를 떠올리게 합니다.

유사성은 무엇보다 하나의 아이디어에 있습니다. 정보나 개념은 일정한 경계 안에서만 의미를 갖는다는 생각입니다.

같은 단어가 서로 다른 두 컨텍스트에서 다른 의미를 가져도, 둘을 반드시 합칠 필요는 없습니다.

컨텍스트는 이 통찰을 다른 영역으로 옮깁니다.

다만 일관된 업무 모델이 아니라 사람의 임무에 따라 경계를 정합니다.

따라서 서로 다른 목표를 추구하는 두 컨텍스트는 같은 주제를 다르게 이해한 내용을 각각 보존할 수 있습니다.

Git: 분기, 자율성, 재통합

Git은 좀 더 직접적인 기술적 영향을 주었습니다.

탐색의 중요성이 충분히 커지면, 안정된 공간을 흔드는 대신 별도의 컨텍스트에서 발전시키는 편이 나을 수 있습니다.

이 방식은 브랜치와 닮았습니다.

탐색은 분기해 고유한 출처와 지식을 만든 뒤, 일부 결론을 원래 공간에 다시 통합할 수 있습니다.

그러나 Git과 마찬가지로 통합한다고 모든 것을 합쳐야 하는 것은 아닙니다.

이 통찰은 컨텍스트 사이의 관계에도 영향을 주었습니다. 지역적 자율성을 유지하고, 서로 달라질 수 있으며, 선택적으로 통합하는 방식입니다.

프로젝트 메모리: 프로젝트의 '왜'를 보존하기

프로젝트 메모리 연구는 컨텍스트를 설계할 때는 제가 알지 못했던 계보적 유사성입니다.

여기서 제기하는 문제는 작업이 몇 달 동안 이어질 때 생기는 문제와 거의 같습니다. 조직은 최종 문서를 비교적 쉽게 보존하지만, 프로젝트가 어떻게, 왜 그런 결과에 도달했는지를 설명하는 맥락은 훨씬 제대로 보존하지 못합니다.

왜 이런 결정을 내렸는가?

그전에 무엇을 시도했는가?

당시 어떤 제약이 있었는가?

새로 온 사람이 계속하기 전에 무엇을 이해해야 하는가?

컨텍스트는 현재 상태, 이력, 결정, 종합, 작업 재개 정보라는 서로 다른 여러 기억으로 같은 질문에 답합니다.

따라서 기능적으로 매우 가깝습니다.

익스피리언스 팩토리: 결과물은 자산이 아니다

빅터 바실리의 연구를 중심으로 발전하고 특히 NASA에서 실험한 익스피리언스 팩토리는 한층 더 중요한 유사성을 보여 줍니다.

프로젝트를 수행하는 조직과, 그 프로젝트의 경험을 다음 프로젝트에서 재사용할 수 있는 지식으로 바꾸는 조직을 구분합니다.

이 아이디어는 컨텍스트의 근본적인 구분 하나와 직접 맞닿아 있습니다.

> 결과물은 자산이 아닙니다.

프로젝트는 훌륭한 결과를 내고도 뒤에 거의 아무것도 남기지 않을 수 있습니다.

반대로 그 경험을 분석하고, 변환하고, 보존하면 다음 문제를 해결하는 역량을 높일 수 있습니다.

컨텍스트는 작업이 이루어지는 바로 그 공간 안에서 경험을 자산으로 바꿉니다.

지식 중심 서비스: 일하면서 지식을 생산하기

지식 중심 서비스(Knowledge-Centered Service), 즉 KCS는 아마 컨텍스트의 가장 흥미로운 이웃 가운데 하나일 것입니다.

그 원칙에 따르면 문서화는 작업이 끝난 뒤 별도로 수행하는 활동이어서는 안 됩니다.

문제를 해결할 때는 기존 지식을 찾아 재사용합니다. 부족하면 개선하고, 없다면 새로 만듭니다.

그렇게 작업 자체가 지식을 만들고 관리하는 작동 방식이 됩니다.

이는 컨텍스트가 추구하는 목표 가운데 하나와 정확히 같습니다.

> 시스템을 사용할 때마다 정보 자산이 늘어날 수 있어야 합니다.

다만 KCS가 주로 서비스와 고객 지원 환경을 위해 설계됐다면, 컨텍스트는 다양한 임무에 적용할 수 있는 일반적인 구조를 제공하려 합니다.

HWMSE: 사람과 기계, 지식이 함께 있는 공간

선행 연구 조사에서 발견한 가장 놀라운 유사성은 아마 중국에서 왔을 것입니다.

메타종합공학 워크숍 홀**(Hall for Workshop of Meta-Synthetic Engineering)**, 즉 HWMSE는 1990년대 초부터 첸쉐썬의 연구를 중심으로 발전했습니다.

복잡한 문제를 다루기 위해 다음 요소를 명시적으로 결합하도록 설계됐습니다.

- 인간 전문가,
- 기계 시스템,
- 지식 시스템입니다.

더 흥미로운 점은 HWMSE 연구가 기존 지식을 사용하는 데서 멈추지 않는다는 것입니다. 지식 창출과, 연구 중인 문제에 고유하면서 사용할수록 계속 풍부해지는 지식 시스템도 다룹니다.

따라서 컨텍스트와 상당히 가깝습니다.

- 문제를 중심으로 한 활동,
- 사람과 기계의 협업,
- 작업 중에 활용하는 자원과 모델,
- 활동을 통해 만들어지는 지식,
- 축적되는 자산이 있습니다.

그렇다고 HWMSE가 컨텍스트와 같은 운영 문법을 가진 것은 아닙니다. 출처, 원자 노트, 상태, 종합, 모순, 작업 재개 정보, 결과물을 구분하는 체계를 이 형태로 찾아볼 수는 없습니다.

그러나 조사한 시스템 가운데 구조적으로 가장 가까운 이웃 중 하나일 것입니다.

ScholOnto와 SWAN: 명제는 출처와 같지 않다

과학적 담론을 다루는 시스템에서도 강한 유사성이 나타납니다.

ScholOnto와 뒤이은 SWAN은 과학 문서와 그 문서에서 만들어진 명제를 구분합니다.

자료 하나가 주장의 기원이 될 수 있습니다.

그 주장은 출처를 보존합니다.

다른 요소가 이를 뒷받침하고, 논의하고, 반박할 수 있습니다.

이는 컨텍스트의 원자 노트를 지탱하는 원칙과 매우 가깝습니다.

> 출처는 출처로 남고, 거기서 만든 지식은 별개의 객체입니다.

이 시스템들은 지식 시스템에 기록된 명제를 자동으로 참이라고 여겨서는 안 된다는 점도 보여 줍니다.

명제에는 출처가 있으며, 추론 안에서 차지하는 지위가 있습니다.

컨텍스트의 인식론적 구조와 특히 강한 유사성을 보이는 부분입니다.

연합형 시스템: 읽기는 복사가 아니다

마지막으로, 컨텍스트 사이의 관계에 관한 일부 아이디어는 연합형 데이터베이스 시스템의 놀랄 만큼 오래된 개념과 이어집니다.

이 구조는 여러 자율적인 데이터베이스가 자율성을 포기하지 않고 협력하게 해야 합니다.

한 시스템이 일부 정보를 외부에 공개할 수 있습니다.

다른 시스템은 그 정보를 원격으로 조회하거나 로컬 사본을 보존하기로 결정할 수 있습니다.

그리고 사본이 만들어지는 순간 고전적인 문제가 나타납니다. 원본이 바뀌면 어떻게 할 것인가?

컨텍스트에서는 바로 이 문제를 다음 문장으로 요약합니다.

> 참조 = 의존 관계, 복사 = 책임.

따라서 컴퓨터 시스템의 작동 방식은 오래전부터 존재했습니다.

컨텍스트는 이를 사람과 AI가 함께 사용하는 공간 사이의 지식 거버넌스 규칙으로 옮깁니다.

무에서 만든 발명이 아니라 종합

이러한 이웃 관계를 보면 컨텍스트의 작동 방식이 모두 완전히 새로운 발명이라고 말하기 어렵습니다.

- 역사가 보여 주는 바가 아닙니다.
- 출처 추적은 이미 존재했습니다.
- 원자 노트도 존재했습니다.
- 프로젝트 기억도 존재했습니다.
- 프로젝트 사이에서 경험을 자산으로 바꾸는 일도 존재했습니다.
- 일하면서 지식을 만드는 방식도 존재했습니다.
- 사람과 기계, 지식 시스템을 결합한 환경도 존재했습니다.
- 출처가 있고 반박할 수 있는 주장도 존재했습니다.
- 자율성을 유지하면서 정보를 공유하거나 복사하는 구조도 존재했습니다.

컨텍스트를 더 잘 특징짓는 것은 이러한 작동 방식을 하나의 방법론 안에서 선택하고, 결합하고, 표준화하는 방식으로 보입니다.

이 방법론은 공통 문법을 부여합니다.

- 출처,

- 원자 노트,
- 모순,
- 결정,
- 상태,
- 이력,
- 종합,
- 행동,
- 작업 재개 정보,
- 결과물,
- 컨텍스트 사이의 관계입니다.

무엇보다 이 조합을 매우 현대적인 문제에 적용합니다. 사람과 AI 사이에서 오래 이어지는 협업을 유지하면서, 모든 대화를 서로 단절되고 기억을 잃은 에피소드로 만들지 않는 문제입니다.

따라서 컨텍스트에는 수많은 선조가 있습니다.

그러나 흔히 그렇듯, 자신의 계보를 알면 무엇이 고유한지를 더 잘 이해할 수 있습니다.