RA

Robo Architect Lite

AI ARCHITECTURE WORKBENCH

제품 소개 및 도입 제안서

요구사항이 바뀌면,
설계도 함께 움직입니다.

요구사항을 글로 입력하면 도메인 주도 설계(DDD)와 인수 조건으로 구체화하고, 프로젝트 설계 원칙에 따라 구현 계획을 작성합니다. 이 PC의 Claude Code가 격리된 작업 공간에서 구현하고, 검증을 거쳐 승인한 변경은 **코드와 설계 그래프에 함께 반영**합니다.



21종
스킬 파일로 정의한 AI 작업 절차

7개
PC에서 실행하는 서비스 컨테이너

Dual Merge
코드와 설계 그래프를 함께 반영

2026년 10월

AI 아키텍처 워크벤치 · 데스크톱 앱(Lite)

uEngine

CONTENTS

목차

01 AI 코딩의 역설 3

02 명세 · 시나리오 · 도메인 4

03 여러 팀의 명세 관리 5

04 Robo Architect Lite 소개 6

05 주요 산출물 ①② 7-8

06 제품 화면 한눈에 보기 9

07 특장점 ①② 요구사항 · 계획 10-11

08 특장점 ③④ 구현 · 검증과 반영 12-13

09 특장점 ⑤⑥ 변경 영향 · 레거시 근거 14-15

10 시스템 구성과 실행 환경 16

11 기술 스택과 스킬 체계 17

12 사용 절차 (실제 화면) 18-19

13 보안 · 거버넌스 20

14 설치 및 납품 21

15 개발 및 품질 검증 체계 22

16 MSAEZ 도입 실적 · 도입 절차 23-24

한 장 요약

AI를 활용한 개발에서는 **구현할 기능과 검증 기준을 먼저 정의하는 일**이 중요합니다. Robo Architect Lite는 요구사항, 업무 시나리오, 도메인 모델을 프로젝트 그래프로 연결합니다. 변경 하나를 프로포절(변경 제안) 하나로 관리하며 설계, 구현, 검증, 반영까지 이어서 진행합니다.

SDD · BDD · DDD

명세로 요구사항을 정의하고, 시나리오로 검증 기준을 정하며, 도메인별로 설계 범위를 구분합니다

프로포절 단위 변경

요구사항부터 계획, 구현, 검증 결과까지 변경 하나의 이력으로 남깁니다

PC에서 실행

그래프 DB와 분석 서비스는 PC의 Docker에서 실행하고, AI 작업은 PC에 설치한 Claude Code가 수행합니다

코드와 설계를 함께 반영하는 변경 관리

코드만 바뀌고 설계 문서가 그대로 남으면 다음 변경을 검토할 때 기준으로 삼기 어렵습니다. Robo Architect Lite는 검증을 마친 프로포절을 승인할 때 **코드 병합과 설계 그래프 반영을 함께 수행**합니다. 그래서 다음 프로포절은 최신 설계를 기준으로 영향 범위를 분석할 수 있습니다.

이 문서가 다루는 것

제품 개요

3–15쪽
배경 · 정의 · 산출물 · 특장점

시스템 구성

16–17쪽
실행 환경 · 기술 스택 · 스킬

사용 절차

18–19쪽
실제 제품 화면 · 6단계

도입 안내

20–24쪽
보안 · 설치 · 실적 · 일정

01 · 배경

AI 코딩의 역설

AI가 코드를 빠르게 작성하더라도 전체 개발 시간이 줄어드는 것은 아닙니다. 생성된 코드를 검토하고 수정하는 시간까지 고려해야 합니다. **요구사항과 업무 맥락, 검증 기준을 명확히 전달하는 것**이 AI 활용의 출발점입니다.

24% 단축
참여 개발자의 사전 예상
AI 사용 시 예상한 작업 시간 감소율

19% 증가
실제 작업 시간
AI 사용 시 작업 완료까지 걸린 시간

16명 · 246건
실험 참여자와 작업 수
숙련 개발자의 기존 오픈소스 프로젝트 작업

출처: METR, 2025년 무작위 대조 실험. 당시 도구와 참여자의 작업 환경에서 측정한 결과입니다.

AI 활용 시 함께 관리해야 할 네 가지 문제

관리 항목 1
업무 맥락 전달

AI가 조직의 업무 규칙과 용어를 이해하려면 관련 정보를 제공해야 합니다. 이 정보가 정리되어 있지 않으면 담당자가 대화마다 다시 설명해야 하고, 누락된 내용은 잘못된 구현으로 이어질 수 있습니다.

관리 항목 2
검토와 수정 부담

코드가 빠르게 생성되어도 요구사항에 맞는지 확인하는 과정은 필요합니다. 검증 기준이 불명확하면 담당자의 판단에 의존하게 되고, 검토와 수정에 드는 시간이 늘어납니다.

관리 항목 3
모호한 요구사항

숙련 개발자는 업무 지식을 바탕으로 기획서의 누락된 내용을 확인합니다. AI에 같은 맥락이 제공되지 않으면 의도와 다른 결과가 나올 수 있으므로, 구현 전에 요구사항을 구체화해야 합니다.

관리 항목 4
설계 근거의 공유와 재사용

설계 결정과 그 근거가 대화 기록에만 남으면 다음 작업에서 찾고 활용하기 어렵습니다. 담당자가 바뀌어도 이어서 사용할 수 있도록 요구사항과 함께 관리해야 합니다.

네 가지 문제에 대한 Robo Architect Lite의 대응

맥락 전달
프로젝트 그래프 · 설계 원칙 · 레거시 분석
AI가 단계마다 같은 업무 맥락을 참조합니다

검토 부담
인수 조건 · 설계 구조 검증
판정마다 근거(파일 · 줄, 통과한 테스트)를 표시합니다

모호한 요구사항
사용자 스토리 · Given · When · Then
구현 전에 구체화하고, 필요하면 선택형 질문으로 확인합니다

설계 근거
프로포절 이력 · 프로젝트 설계 원칙
결정과 근거를 기록해 다음 변경에서 다시 사용합니다

AI가 작성할 코드의 요구사항과 검증 기준을 먼저 정해야 합니다. 업무 규칙과 설계 근거를 함께 관리하면, 구현과 검토에 필요한 맥락을 일관되게 제공할 수 있습니다.

02 · 배경

명세 · 시나리오 · 도메인으로 맥락을 구체화합니다

Robo Architect는 세 가지 방법론을 연결합니다. **명세 주도 개발(SDD)**로 요구사항을 정의하고, **행동 주도 개발(BDD)**로 기대 동작을 구체화하며, **도메인 주도 설계(DDD)**로 업무 영역과 용어의 경계를 정합니다.

SDD · 명세 주도 개발

구현할 내용을 명세로 정의

목적과 요구사항을 문서로 정리한 뒤, 구현 계획과 작업 목록을 작성합니다. AI와 개발자가 같은 명세를 기준으로 작업할 수 있도록 합니다.

BDD · 행동 주도 개발

기대 동작을 사례로 설명

"잔액이 100원일 때 20원을 인출하면 잔액은 80원이 된다" 처럼 조건, 행동, 결과를 구체적으로 적습니다. 이 시나리오를 검증 기준으로 사용합니다.

DDD · 도메인 주도 설계

업무 영역별 설계 범위 구분

업무 개념과 규칙을 기준으로 도메인의 경계를 정합니다. 각 영역에서 사용하는 용어와 책임을 명확히 해 AI에 필요한 맥락을 전달합니다.

현금 인출 요구사항을 구체화하는 예

작성 방식	예시 — ATM 현금 인출	검토 시 확인할 점
사용자 스토리	"고객은 계좌에서 현금을 인출할 수 있어야 한다"	잔액 부족 등 예외 상황과 처리 조건을 추가해야 합니다
요구사항 명세서	"시스템은 인증된 사용자에게 현금 인출 기능을 제공한다"	인증 조건과 인출 한도, 실패 시 동작을 구체화해야 합니다
상세 설계서	`if balance >= amount: balance -= amount`	처리 로직을 표현하지만, 비개발자가 검토하기는 어렵습니다
BDD 시나리오	**Given** 잔액 100원 · **When** 20원 인출 · **Then** 현금 20원 지급, 잔액 80원	업무 담당자와 개발자가 같은 사례로 기대 결과를 확인합니다

Robo Architect Lite에서 세 방법론이 쓰이는 위치

SDD
프로포절 · Plan · 작업 목록
구현 전에 명세와 계획을 확정합니다

BDD
Given · When · Then 인수 조건
검증 단계의 판정 기준이 됩니다

DDD
컨텍스트 · Aggregate · 명령 · 이벤트
상세 DDD 방식은 6단계로 설계합니다

BDD 시나리오는 업무 규칙을 함께 검토하는 기준입니다. 기획자와 고객은 조건과 결과가 업무 의도에 맞는지 확인하고, AI는 이를 테스트 작성의 근거로 활용합니다. Robo Architect는 시나리오를 관련 설계 요소와 연결해 그래프에 저장하고, 구현 후 검증 단계에서 판정 기준으로 사용합니다.

03 · 배경

여러 팀의 명세를 연결해 관리해야 합니다

파일 기반 명세는 팀의 요구사항과 구현 계획을 정리하는 데 유용합니다. 다만 **여러 팀과 저장소에 정보가 분산되면** 용어를 맞추고 변경의 영향 범위를 파악하기 어려워집니다. 조직 차원에서는 다음 네 가지 문제를 함께 관리해야 합니다.

관리 과제	발생 원인	업무에 미치는 영향
분산된 명세	명세 파일이 각 저장소 안에만 존재합니다	다른 팀의 구현을 파악하기 어려워 같은 기능을 중복 개발할 수 있습니다
용어 불일치	저장소마다 도메인 용어를 각자 정의합니다	"수용가"와 "고객"처럼 서로 다른 용어가 같은 대상을 뜻하는지 확인해야 합니다
영향 범위 파악	요구사항 · 설계 · 코드의 관계를 문서 링크로 관리합니다	규정이 바뀌면 관련 문서와 코드를 일일이 확인해야 합니다
설계 지식의 인계	업무 지식과 설계 근거가 담당자와 대화 기록에 의존합니다	담당자 변경 시 업무 지식이 누락되거나 같은 분석을 반복할 수 있습니다

> Robo Architect는 **요구사항 · 도메인 모델 · 인수 조건 · 변경 이력의 관계**를 프로젝트 그래프로 관리합니다. 새 변경을 제안하면 연결된 설계 요소를 따라 새로 만들 요소와 수정할 요소를 구분하고, 이미 구현된 요소와의 충돌 가능성을 표시합니다.

파일 기반과 그래프 기반의 차이

비교 항목	파일 기반 명세	그래프 기반 명세
명세의 위치	저장소 안의 마크다운 파일	프로젝트마다 하나씩 두는 도메인 그래프
경계의 기준	저장소와 폴더 단위	Bounded Context 단위의 업무 영역
용어 일관성	저장소마다 각자 기술	도메인별 용어와 관계를 함께 관리
변경 영향 분석	파일 검색과 담당자 협의로 확인	연결 관계를 따라 영향 범위와 충돌 가능성 분석
검증 기준	문서에 적은 수용 기준	설계 요소에 연결된 Given · When · Then 인수 조건
지식의 축적	프로젝트별 파일에 기록	설계 요소와 프로포절(변경 제안) 이력으로 연결해 보존

파일 기반 도구와 그래프 기반 관리는 함께 활용할 수 있습니다. 팀은 기존 Git 저장소에서 코드를 관리하고, Robo Architect는 설계 요소 간 관계와 변경 이력을 그래프로 관리합니다. 구현 코드는 대상 저장소의 브랜치에 커밋으로 남습니다.

04 · 소개

Robo Architect Lite 소개

Robo Architect Lite는 **명세 주도 개발을 위한 데스크톱 AI 아키텍처 워크벤치**입니다. 요구사항을 DDD 설계와 인수 조건으로 구체화하고 설계 원칙에 따라 계획을 작성합니다. 구현은 Claude Code가 격리된 작업 공간에서 수행합니다.

입력

요구사항

추가하거나 변경할 기능을 문장으로 입력하거나 요구사항 문서를 붙여 넣습니다. 기존 코드와 데이터베이스를 연결해 설계 근거로 활용할 수 있습니다.

처리

DDD 설계와 구현 계획

바운디드 컨텍스트, 기능, 사용자 스토리, 인수 조건으로 구체화합니다. 설계 원칙에 따라 Aggregate · 명령 · 이벤트와 아키텍처를 결정합니다.

출력

코드와 설계 그래프

전용 브랜치에 구현한 코드와 검증 결과를 제공합니다. 승인하면 대상 브랜치와 프로젝트 그래프에 함께 반영합니다.

기존 방식과의 차이

	문서 중심의 설계 관리	Robo Architect Lite
설계의 형태	다이어그램 이미지와 문서	질의할 수 있는 프로젝트 그래프(노드 26종 · 관계 32종 정의)
변경 단위	티켓과 문서를 따로 관리	프로포절 하나에 요구사항 · 계획 · 구현 · 검증 결과를 함께 기록
설계 기준	담당자가 문서로 관리	프로젝트 설계 원칙을 계획의 결정마다 근거로 표시
구현	설계와 별도로 진행	전용 브랜치의 격리된 작업 공간에서 Claude Code가 구현
검증	리뷰어의 판단에 의존	인수 조건과 설계 구조를 근거(파일 · 줄)와 함께 판정
AI 작업 절차	사람마다 다른 프롬프트	스킬 파일 21종에 절차와 규칙을 정의

한눈에 보는 흐름

INTENT 요구사항 구체화 → **PLAN** 설계 원칙 · 계획 → **IMPLEMENTING** 격리된 구현 → **VALIDATING** 구조 · 인수 조건 → **ACCEPTED** 코드 + 그래프 반영

사용자는 단계별 결과를 검토한 뒤 다음 단계로 진행합니다. **대상 브랜치와 설계 그래프에는 최종 승인(Accept) 후에 반영**됩니다.

05 · 산출물

주요 산출물 ① — 프로젝트 설계 그래프

프로포절의 설계 결과는 **프로젝트 그래프**에 저장됩니다. 프로젝트 하나가 그래프 DB(Ontological DB) 안의 그래프 하나이므로, 프로젝트를 여러 개 만들어도 설계가 섞이지 않습니다.

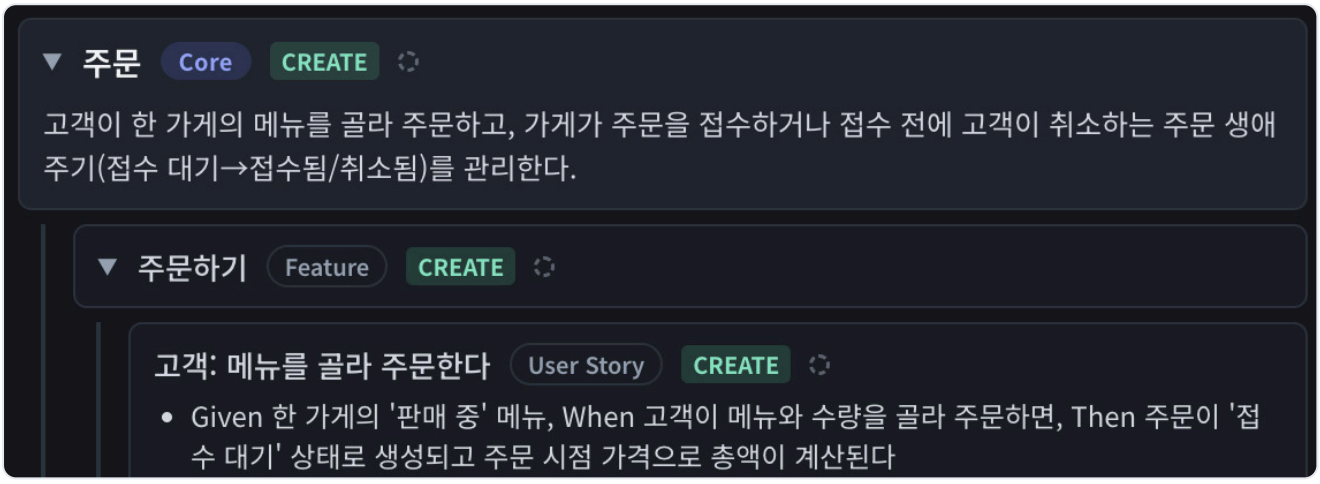


그림 1. 전략 설계 — 바운디드 컨텍스트 「주문」(Core) 아래의 기능 「주문하기」, 사용자 스토리, Given · When · Then 인수 조건. 요소마다 신규(CREATE)인지 표시합니다. *데모 데이터(튜토리얼 실습) 기준입니다.*

그래프에 저장하는 정보

계층	노드
요구사항	Requirement · Feature · UserStory · Given/When/Then(인수 조건)
전략 설계	BoundedContext(Core · Supporting · Generic 분류) · Policy
전술 설계	Aggregate · Command · Event · ReadModel · Property · Invariant
화면 · 흐름	UI · Journey · JourneyStep
변경 · 기준	Proposal(PRO-NNN) · Constitution(프로젝트 설계 원칙)

26종
노드 유형
요구사항부터 프로포절 · 설계 원칙까지

32종
관계 유형
EMITS · TRIGGERS · HAS_GIVEN · EFFECT …

1 : 1
프로젝트와 그래프
프로젝트마다 그래프를 따로 생성

설계 요소의 관계를 따라 필요한 정보를 찾습니다. 요구사항 탭에서 사용자 스토리의 「설계 궤적」을 열면 주문하기 화면 → PlaceOrder(명령) → Order(Aggregate) → OrderPlaced(이벤트)처럼 스토리를 처리하는 설계 요소를 한 줄로 확인할 수 있습니다.

05 · 산출물

주요 산출물 ② — 설계 원칙 · 계획 · 코드 · 검증 기록

프로포절은 단계마다 산출물을 남깁니다. 나중에 "이 Aggregate는 왜 생겼는가?"를 확인해야 할 때 해당 프로포절을 열어 **요구사항, 계획의 근거, 구현 커밋, 검증 결과**를 함께 볼 수 있습니다.

단계	산출물	보관 위치
INTENT	전략 설계(프로세스 · 바운디드 컨텍스트 · 기능 · 사용자 스토리 · 인수 조건), Impact Map	프로포절
설계 원칙	Core Principles · Technology Constraints · Architecture · Repository Strategy · Governance로 구성한 Markdown 문서	프로젝트 그래프(프로젝트당 1개)
PLAN	전술 설계, 아키텍처 결정과 근거 원칙, 원칙이 정하지 않은 영역, 영향 분석	프로포절
구현	Phase별 작업 목록, 작업마다 1건씩 기록하는 커밋	대상 저장소의 `proposal/PRO-NNN` 브랜치
검증	판정 카드(PASS · FAIL · SKIP)와 근거(파일 · 줄, 통과한 테스트)	프로포절
ACCEPTED	병합된 코드, 갱신된 설계 그래프, 상태 변경 이력	대상 브랜치 · 프로젝트 그래프

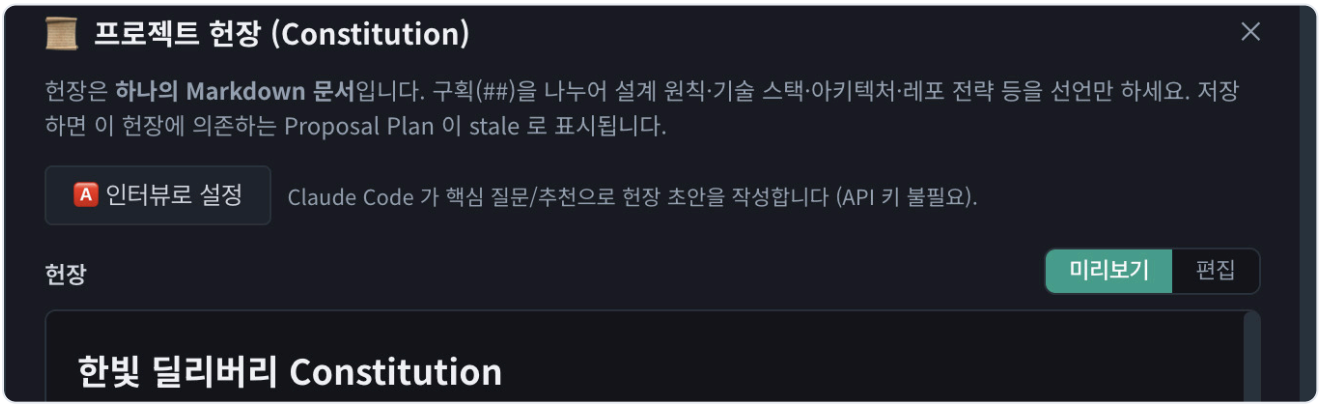


그림 2. 프로젝트 설계 원칙(Constitution) 편집기 — 인터뷰로 작성한 「한빛 딜리버리 Constitution」을 미리보기 · 편집합니다. *데모 데이터(튜토리얼 실습) 기준입니다.*

튜토리얼 실습 프로포절 PRO-001의 결과(데모 데이터)

2 → 7
질문 → 설계 원칙
인터뷰 질문 2개로 원칙 7개 작성

15건
작업 · 커밋
Phase 6개, 작업마다 커밋 1건

69개 파일
3,622줄
잠금 파일 제외

45건
테스트 통과
병합 후 `npm test` 실행 결과

06 · 화면

제품 화면 한눈에 보기

홈의 기능 카드 여덟 개가 각 화면을 탭으로 엽니다. 요구사항 · 프로세스 · 설계 · 데이터 탭은 **같은 프로젝트 그래프를 용도에 맞게** 보여 줍니다. 아래는 실제 제품 화면이며, 데모 데이터(튜토리얼 실습) 기준입니다.

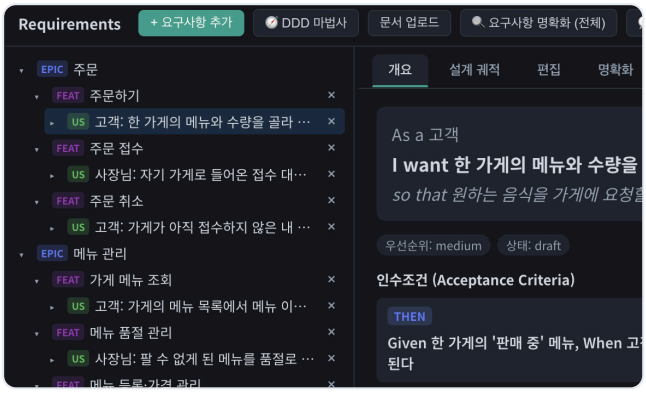


그림 3. 요구사항 — 컨텍스트(EPIC) › 기능(FEAT) › 스토리(US)와 인수 조건

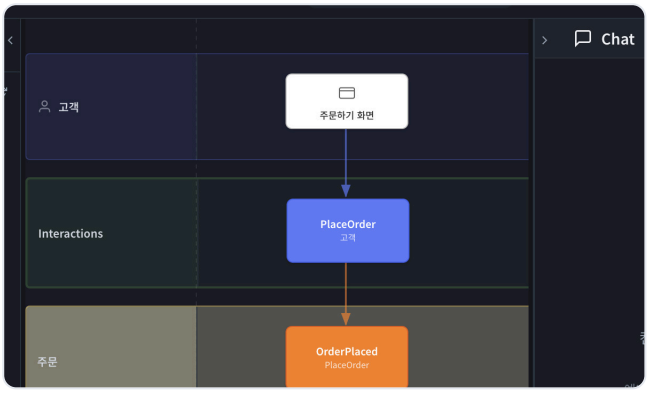


그림 4. 프로세스 — 이벤트 모델링: 주문하기 화면 → PlaceOrder → OrderPlaced



그림 5. 설계 캔버스 — 화면 → 명령(파랑) → Aggregate(노랑) → 이벤트(주황)

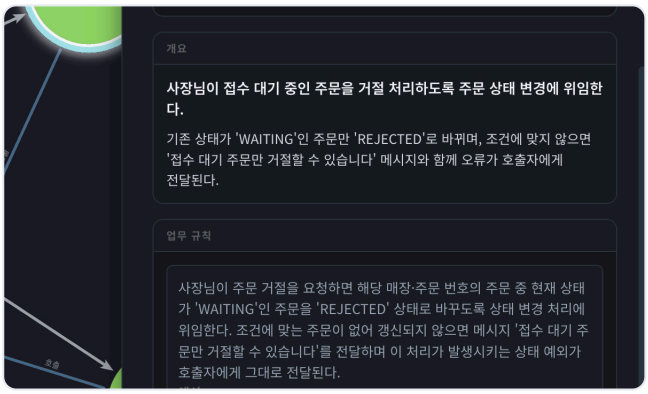


그림 6. 레거시 — 코드 그래프와 메서드별 개요 · 업무 규칙 · 예시

홈의 기능 카드

카드	하는 일	카드	하는 일
프로포절	요구사항 → 설계 · 계획 · 구현 · 검증	**데이터 모델**	Aggregate의 엔티티 · 값 객체 · 열거형
요구사항	컨텍스트 › 기능 › 스토리 › 인수 조건	**코드**	코드 폴더의 파일과 Claude Code 터미널
프로세스	BPM 흐름과 이벤트 모델링	**레거시 · 데이터 소스**	기존 코드 분석, 기존 DB 연결
설계 캔버스	이벤트스토밍 — Aggregate · 명령 · 이벤트	**프로젝트 헌장**	프로젝트 설계 원칙 편집 · 인터뷰

07 · 특장점

① 요구사항을 설계와 인수 조건으로 구체화합니다

문장으로 쓴 요구사항에는 예외 상황과 경계 조건이 빠지기 쉽습니다. 사용자가 요구사항을 입력하면 Claude Code가 INTENT 스킬에 따라 **프로세스 · 바운디드 컨텍스트 · 기능 · 사용자 스토리 · 인수 조건**으로 구체화합니다. 사용자는 문장에 없던 경계 상황까지 인수 조건으로 확인할 수 있습니다.



그림 7. PRO-001의 전략 설계 — 요구사항 문장에서 프로세스 두 개를 도출했습니다. *데모 데이터(튜토리얼 실습) 기준입니다.*

결과(132초)	내용
프로세스 2개	메뉴 판매 운영 · 주문 생성과 접수 전 취소
컨텍스트 2개	주문(Core) · 메뉴 관리(Supporting)
기능 6개	메뉴 등록 · 가격 관리, 품절 관리, 주문하기 등
스토리 7개	사장님 4개 · 고객 3개, 스토리마다 인수 조건

문장에 없던 경계 조건까지 제시합니다

- **Given** '접수 대기' 상태의 내 주문 · **When** 고객이 취소하면 · **Then** '취소됨'이 되고 가게는 그 주문을 접수할 수 없다
- **Given** 가게가 이미 접수한 주문 · **When** 고객이 취소하면 · **Then** 취소가 거부되고 그 이유가 안내된다
- **Given** 이미 취소된 주문 · **When** 다시 취소하면 · **Then** 상태 변화 없이 이미 취소된 주문임이 안내된다

변경 범위에 맞는 세 가지 방식

방식	진행 방법
간소화	INTENT → PLAN 두 단계. 영향 범위가 작은 변경에 적합합니다
상세 DDD	Discover → Decompose → … → Tactical 6단계를 하나씩 확정합니다
ODA 표준	TM Forum ODA 표준을 근거로 설계하고 적합성을 점검합니다

요구사항 문서로도 시작할 수 있습니다. 다섯 구획의 문서를 붙여 넣자 컨텍스트 5개, 기능 10개, 사용자 스토리 17개, 프로세스 3개가 나왔습니다(171초).

07 · 특장점

② 프로젝트 설계 원칙을 근거로 구현 계획을 작성합니다

같은 기능도 기술 스택과 아키텍처 스타일에 따라 구현 계획이 달라집니다. 이런 결정을 프로포절마다 새로 하면 프로포절끼리 서로 다른 구조를 만들게 됩니다. Robo Architect는 프로젝트마다 **설계 원칙 문서(Constitution)**를 하나 두고, 모든 계획이 이 원칙을 따르도록 합니다.

질문 두 개로 작성한 설계 원칙 일곱 개

질문	추천 근거	선택한 답
아키텍처 스타일	바운디드 컨텍스트가 적고 단순해 단일 배포가 간단합니다	MONOLITH(추천)
기술 스택	DDD Aggregate와 트랜잭션 일관성에 적합한 스택을 추천합니다	직접 입력: Node.js(Express) · React · SQLite

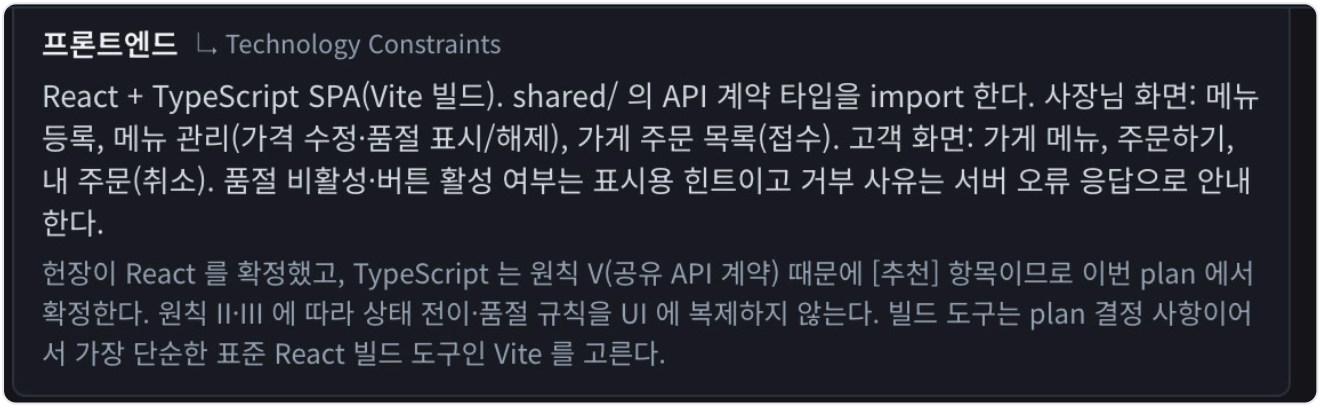


그림 8. Plan의 아키텍처 결정 「프론트엔드」 — 근거가 된 설계 원칙 구획(↳ Technology Constraints)과 판단 이유를 함께 표시합니다. *데모 데이터(튜토리얼 실습) 기준입니다.*

결정	내용	근거 원칙
배포 환경	Node.js 프로세스 1개, SQLite 단일 파일 DB, React 빌드를 같은 앱이 제공	Architecture
데이터 접근	모듈별 저장소 계층, 접수 · 취소는 상태 조건부 갱신으로 하나만 성공	II · Technology Constraints
테스트 전략	도메인 불변식 단위 테스트와 API 통합 테스트	VI

> **원칙이 정하지 않은 영역은 따로 표시합니다.** 이번 계획에서는 "어느 사장님이 어느 가게의 주인인가"를 정한 원칙이 없어 「헌장 공백」으로 표시됐습니다. 설계 원칙을 수정하면 기존 계획은 「오래됨」으로 표시되며, 최신 원칙으로 확정한 계획이 있어야 구현을 시작할 수 있습니다.

08 · 특장점

③ 격리된 작업 공간에서 Claude Code가 구현합니다

AI가 작성한 코드가 기존 코드에 바로 섞이면 검토 범위를 정하기 어렵습니다. Robo Architect는 프로포절마다 **전용 브랜치(`proposal/PRO-001`)의 Git 워크트리**(같은 저장소의 별도 작업 폴더)를 만들고 그 안에서만 구현합니다. 대상 브랜치는 최종 승인 전까지 변경되지 않습니다.

1. **작업 목록 생성** — 확정된 Plan을 Phase별 작업으로 나누고, 작업마다 생성할 파일 경로를 붙입니다.
2. **구현 시작** — 대상 프로젝트 폴더 안에 워크트리와 전용 브랜치를 만들고, 코드 탭에 프로포절 전용 Claude Code 셀을 엽니다.
3. **작업별 커밋** — Claude Code가 `/robo-implement PRO-001`로 작업을 하나씩 수행하고, 작업을 마칠 때마다 체크 표시와 커밋을 남깁니다.
4. **진행 확인 · 개입** — 진행 막대와 Phase별 완료 수를 확인하고, 셀에서 작업을 멈추고 추가 지시를 입력합니다.

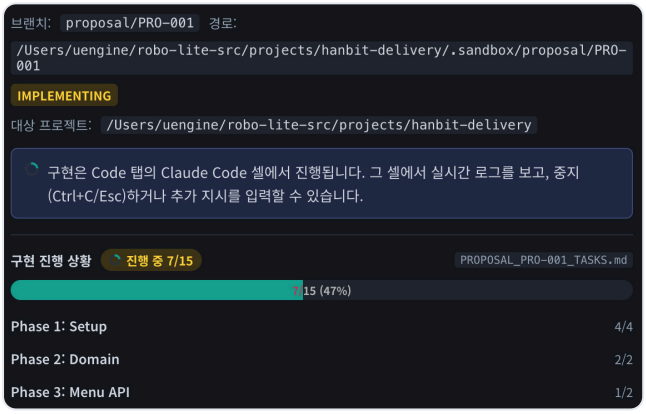


그림 9. 구현 중인 PRO-001 — 브랜치 · 워크트리 경로, 진행 7/15. *데모 데이터(튜토리얼 실습) 기준입니다.*

Phase	작업
1 Setup	저장소 구성, Express · SQLite 기반, 인증, API 계약
2 Domain	Menu · Order Aggregate와 불변식
3 Menu API	메뉴 저장, 공개 인터페이스, 라우트
4 Ordering API	주문 생성 · 접수 · 취소
5 Frontend	사장님 화면 · 고객 화면
6 Test	불변식 테스트 · API 통합 테스트

15개
작업
Phase 6개

약 22분
구현 시간
확인 창 응답 이후

15건
커밋
작업마다 1건

3,622줄
69개 파일
잠금 파일 제외

구현 결과 — Plan에서 정한 모듈 경계대로 생성된 폴더

`backend/src/modules/menu` · `ordering` · `backend/migrations` · `frontend/src/pages/owner` · `customer` · `shared/src`

Claude Code의 안전 확인은 사용자가 직접 응답합니다. 처음 실행할 때 Claude Code가 폴더 신뢰와 권한 우회 모드 동의를 묻고, Robo Architect는 이 확인에 대신 응답하지 않습니다. 권한 우회는 격리된 작업 공간에서만 작업하는 구현 셀을 위한 설정입니다.

08 · 특장점

④ 설계와 대조해 검증하고, 코드와 설계를 함께 반영합니다

작업 목록을 모두 완료했다고 해서 설계대로 구현됐다고 볼 수는 없습니다. 검증 단계는 두 가지를 확인합니다. **구조 검증**은 전술 설계의 Aggregate · 필드 · 명령 · 이벤트가 코드에 있는지 robo-sync 추출기로 대조합니다. **인수 조건 검증**은 Given · When · Then마다 코드와 테스트를 확인하고 근거를 붙입니다.



구분	건수	PASS	FAIL	SKIP
인수 조건	18	18	0	0
구조 검증	31	26	0	5
합계	49	44	0	5 (통과율 90%)

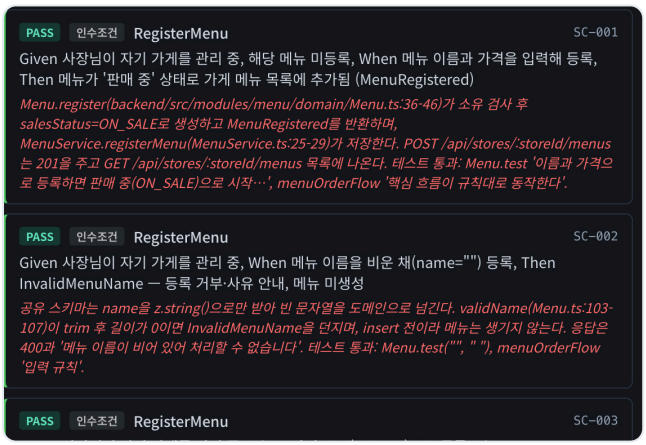


그림 10. 인수 조건 카드 — 판정, 대상 명령, 시나리오 번호와 근거(파일 · 줄, 통과한 테스트). *데모 데이터(튜토리얼 실습) 기준입니다.*

SKIP은 자동 대조를 하지 못한 항목입니다. 이번 SKIP 5건은 명령이 별도 타입 없이 서비스 메서드 인자로만 표현되어 추출기가 입력 필드를 읽지 못한 경우였습니다. 통과율은 SKIP을 포함한 전체 건수로 계산합니다.

FAIL이 있으면 확인 절차를 거칩니다. 「실패 항목을 인지하고 Accept를 진행합니다」를 선택해야 승인할 수 있으며, 그 기록이 남습니다.

승인과 되돌리기

작업	처리 내용
Accept (Dual Merge)	전용 브랜치를 대상 브랜치에 병합하고, 전략 · 전술 설계를 프로젝트 그래프에 반영합니다. 그래프 반영이 실패하면 코드 병합을 되돌리고 실패 상태로 표시합니다
수거(되돌리기)	「그래프만 되돌리기」 또는 「그래프 + 코드 되돌리기」(병합 커밋 `git revert`)를 선택합니다
Destroy	승인하지 않고 폐기합니다. 격리된 작업 공간을 정리하고, 프로포절의 변경 이력은 보관합니다

09 · 특장점

⑤ 변경이 기존 설계에 미치는 영향을 구분해 보여 줍니다

실제 개발은 대부분 기존 설계를 수정하는 일입니다. 승인된 설계가 있는 프로젝트에서 새 프로포절을 만들면, Robo Architect는 **새로 만드는 요소(CREATE)와 기존 요소를 수정하는 요소(MODIFY)**를 구분합니다. 이미 구현된 요소를 수정하는 경우에는 충돌 가능성을 함께 표시합니다.

요구사항 입력 (주문 거절 사유) → **INTENT** (기존 컨텍스트에 추가) → **PLAN** (기존 Aggregate 수정) → **Impact Map** (신규 · 수정 · 충돌)

두 번째 프로포절 PRO-002에 "사장님은 주문을 거절할 때 사유를 하나 고르고, 고객은 그 사유를 본다"를 입력했습니다. INTENT 분석(80초)에서는 새 바운디드 컨텍스트 없이 기존 「주문」 컨텍스트에 기능 두 개가 추가됐습니다. Plan 생성(약 6분)에서는 PRO-001에서 확정한 결정(데이터 접근 방식, 테스트 도구, 폴더 구성)을 그대로 이어받았습니다.

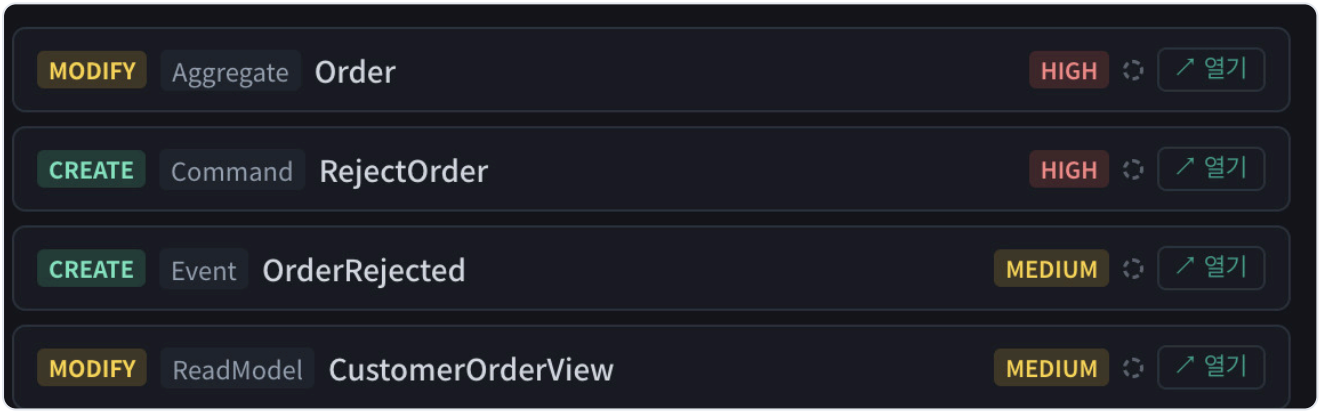


그림 11. PRO-002의 전술 설계 — 기존 Order와 CustomerOrderView는 MODIFY, RejectOrder와 OrderRejected는 CREATE로 표시됩니다. *데모 데이터(튜토리얼 실습) 기준입니다.*

연산	요소	충돌 가능성	변경 내용
MODIFY	Aggregate Order	**HIGH**	상태에 REJECTED 추가, 거절 사유 열거형 · 속성, 불변식 2개
CREATE	Command RejectOrder	NONE	사유를 받아 주문 거절
CREATE	Event OrderRejected	NONE	주문 거절됨
MODIFY	ReadModel CustomerOrderView	NONE	고객 주문 조회에 거절 사유 추가

HIGH는 진행을 막는 표시가 아니라 검토할 위치를 알려 주는 표시입니다. Order는 이미 구현 · 승인된 Aggregate이므로 검증 단계에서 기존 인수 조건이 계속 통과하는지 확인해야 합니다. 같은 설계 요소를 다른 프로포절이 구현하고 있으면 Plan 단계로 제출할 수 없도록 알립니다.

09 · 특장점

⑥ 기존 코드와 데이터베이스를 설계 근거로 활용합니다

기존 시스템의 업무 규칙은 문서보다 코드에 정확히 남아 있는 경우가 많습니다. Robo Architect는 레거시 코드를 분석해 **클래스 · 메서드 · 테이블과 업무 규칙**을 그래프로 구조화합니다. 새 프로포절을 만들면 요소마다 근거가 된 기존 코드를 연결합니다.

1 DDL 파싱 — 2 소스 적재 — 3 프로젝트 스캔 — 4 관계 연결 — 5 분석 맥락 구성 — 6 코드 분석(AI) — 7 후처리

141초	6 · 18 · 7	53개	11 / 11
분석 시간	**클래스 · 메서드 · 테이블**	**업무 규칙**	**근거 연결**
Java · DDL 예제 시스템	컬럼 32개, 호출 46개	실행마다 다름(35개 · 53개)	근거 15건, 새 요소 11개 모두 연결

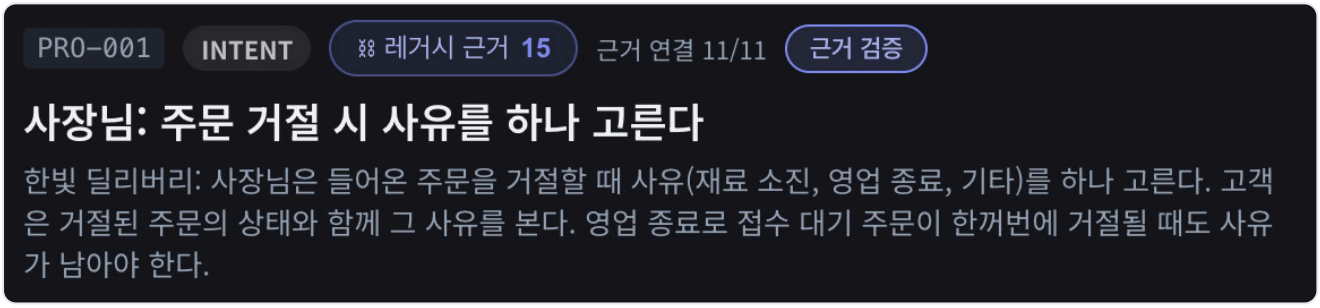


그림 12. 레거시 코드를 근거로 만든 프로포절 — 머리에 「레거시 근거 15 · 근거 연결 11/11」을 표시합니다. *데모 데이터(튜토리얼 실습) 기준입니다.*

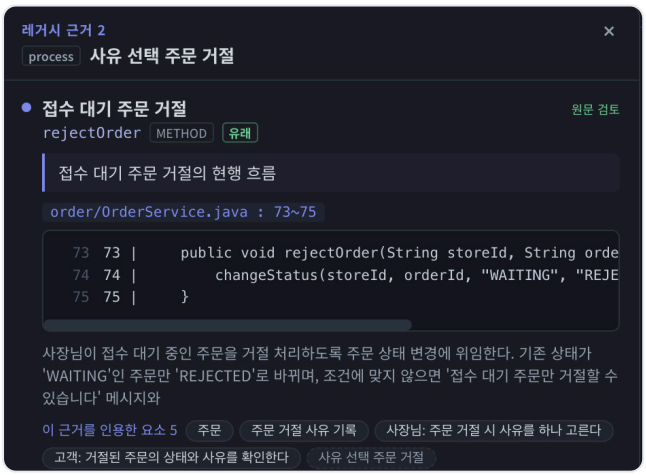


그림 13. 프로세스 「사유 선택 주문 거절」의 근거 — `order/OrderService.java : 73~75` 원문과 이 근거를 인용한 요소 5개. *데모 데이터(튜토리얼 실습) 기준입니다.*

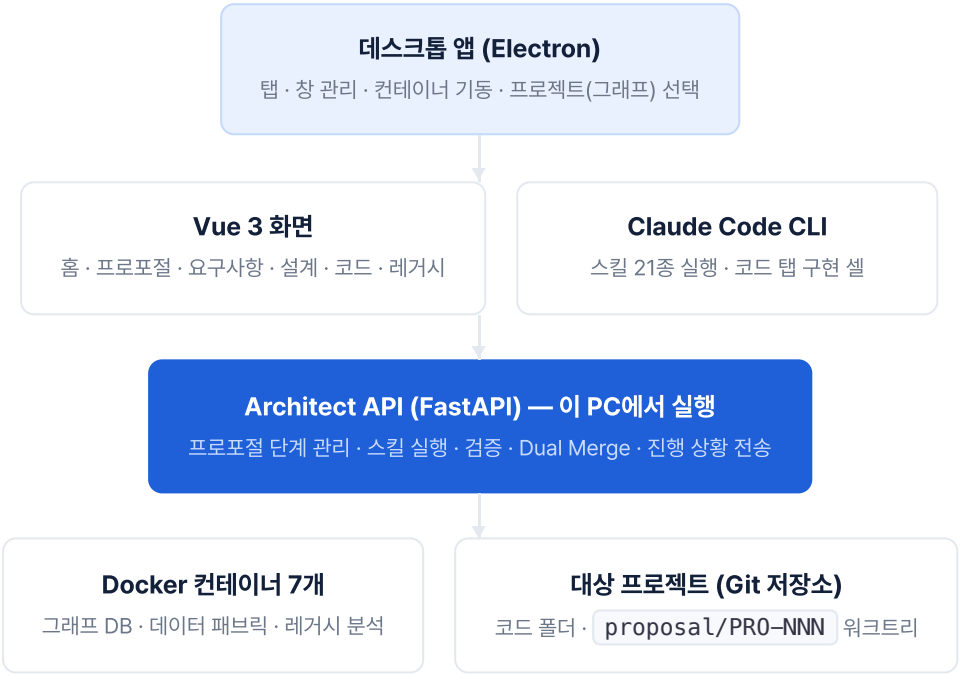
표시	뜻
유래 · 정제	기존 동작에서 나온 요소 / 기존 동작을 수정하는 요소
원문 검토	원문 코드까지 열어 확인한 근거
검색 확인	검색 결과로 확인한 근거
파일 : 줄	근거가 된 원문 위치와 코드
인용한 요소	같은 코드에 기대는 새 요소 목록

요구사항 문장에 없던 조건도 근거에서 도출합니다. 기존 시스템이 거절 사유를 저장하지 않는다는 사실에서 "사유 기록 기능 도입 이전에 거절된 주문은 '사유 정보 없음'으로 표시한다"는 인수 조건이 나왔습니다. 기존 DB는 데이터 소스(PostgreSQL · MySQL · MariaDB)로 등록해 실제 테이블을 확인할 수 있습니다.

10 · 아키텍처

시스템 구성과 실행 환경

Robo Architect Lite는 **데스크톱 앱, 이 PC에서 실행하는 Architect API, Docker 컨테이너 7개**로 구성됩니다. 앱을 실행하면 앱이 컨테이너를 직접 기동하고, 앱을 종료해도 컨테이너를 유지해 다음 실행 시간을 줄입니다. AI 작업은 이 PC의 Claude Code가 수행합니다.



컨테이너 구성

컨테이너	구분	역할
graph-db **graph-bolt**	Ontological DB	설계 그래프와 분석 그래프를 저장하며, 프로젝트마다 그래프가 하나씩 생깁니다. graph-bolt는 Bolt 프로토콜 입구로, 상태 없이 질의를 그래프 엔진에 전달합니다
fabric **catalog**	데이터 패브릭	등록한 기존 DB를 그래프 DB에 연결하고, 테이블 · 컬럼 · 관계 정보를 제공합니다
analyzer **parser** **gateway**	레거시 분석	구문 분석(ANTLR)과 코드 의미 · 업무 규칙 분석을 수행합니다. 분석기의 AI 호출은 Architect API를 거쳐 Claude Code가 처리합니다

설계 그래프와 분석 그래프를 분리합니다

레거시 분석은 분석용 그래프를 비우고 다시 작성합니다. 설계 그래프와 이름이 같으면 그래프 DB가 기동을 거부하도록 구성해, 분석 작업이 설계 데이터를 지우는 일을 막습니다.

11 · 아키텍처

기술 스택과 스킬 체계

고객사 개발팀이 구조를 이해하고 유지보수할 수 있도록 **널리 사용되는 프레임워크와 오픈소스 기술**로 구성했습니다. AI의 작업 절차는 제품 코드와 분리해 스킬 파일로 관리합니다.

레이어	구성
데스크톱	Electron 33 · electron-builder 25(Windows NSIS · macOS DMG) · TypeScript
프런트엔드	Vue 3.5 · Vite 5 · Pinia · Vue Flow(캔버스) · bpmn-js 18 · Mermaid 11 · Tailwind CSS 4 · CodeMirror 6 · xterm.js 6
백엔드	Python 3.11 이상 · FastAPI · Uvicorn · Pydantic v2 · SSE(서버 전송 이벤트) · MCP
그래프 DB	Ontological DB — 그래프 엔진과 Bolt 게이트웨이, Cypher 질의
AI 실행	Claude Code CLI(스킬 실행 · 코드 탭 PTY 셸) · 분석기용 OpenAI 호환 연결 모듈
분석 서비스	레거시 분석기 · ANTLR 구문 분석기 · 데이터 패브릭 · 데이터 카탈로그 · API 게이트웨이
검증	pytest · Playwright · ruff

스킬 파일 21종

묶음	스킬
robo-proposals 15종	단계 — `intent` · `project-constitution` · `plan` · `tasks` · `implement` · `test` · `context`(영향 분석) 상세 DDD — `scope` · `discover` · `decompose` · `strategize` · `connect` · `define` · `tactical` 표준 — `oda`
robo-spec 4종	`robo-implement`(코드 탭 셀의 구현) · `robo-sync`(코드 → 설계 구조 대조) · `robo-plan` · `robo-tasks`
robo-changes 2종	`robo-change-specify` · `robo-change-tasks`(요구사항 변경)

스킬 파일을 수정하면 제품 코드를 바꾸지 않고 판단 기준을 바꿀 수 있습니다. 튜토리얼에서 INTENT 스킬에 "바운디드 컨텍스트 이름에 영문 이름을 함께 표기한다"는 규칙 한 줄을 추가했습니다. 앱을 다시 시작하지 않아도 다음 분석부터 「메뉴 관리(MenuManagement)」·「주문(Ordering)」으로 나왔습니다.

12 · 사용 절차

사용 절차 — 여섯 단계

프로젝트를 열고 프로포절을 작성한 뒤, 설계와 계획을 검토하고 구현 · 검증 · 승인을 진행합니다. 아래는 **실제 제품 화면으로 보는 여섯 단계의 사용 절차**입니다. 화면은 데모 데이터(튜토리얼 실습) 기준입니다.

1 프로젝트 열기 — 2 프로포절 작성 — 3 전략 설계 검토 — 4 설계 원칙 · 계획 — 5 격리된 구현 — 6 검증 · 승인

1단계 · 프로젝트 만들기와 열기

홈에서 새 프로젝트 이름(예: `hanbit_delivery`)을 입력하고 코드 폴더로 사용할 Git 저장소를 선택합니다. 「프로젝트 열기」를 누르면 앱이 이 프로젝트의 그래프로 백엔드를 연결하고 기능 카드를 활성화합니다.



그림 14. 프로젝트 선택 — 프로젝트마다 그래프가 따로 생기며, 다른 그래프와 섞이지 않습니다.

2단계 · 프로포절 작성

프로포절 탭에서 「+ 새 Proposal」을 누르고, 추가하거나 변경할 기능을 문장으로 입력합니다. 진행 방식(간소화 · 상세 DDD · ODA 표준)을 선택하고 「AI 분석 시작」을 누릅니다.

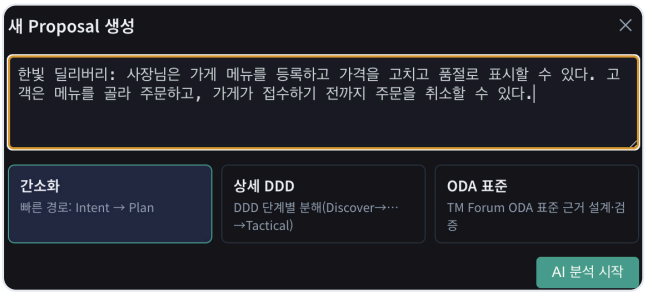


그림 15. 새 Proposal 생성 창

3단계 · 전략 설계 검토

진행 창에 Claude Code의 작업 내용이 표시됩니다. 분석이 끝나면 전략 설계와 Impact Map을 검토하고, 필요하면 피드백을 적어 다시 생성합니다.



그림 16. 분석 진행 창 — 요구사항마다 레거시 그래프를 검색합니다.

AI가 확인할 사항이 있으면 질문합니다

요구사항이 모호하면 분석 중에 선택형 질문이 나옵니다. 보기를 고르거나 직접 입력해 답하고, 질문 없이 진행하려면 「현재 정보로 계속」을 누릅니다.

12 · 사용 절차

4–6단계 · 계획 · 구현 · 검증과 승인

4단계 · 설계 원칙과 계획 확정

「Plan 단계로 진행」 후 「Plan 생성」을 누릅니다. 설계 원칙이 없는 프로젝트라면 인터뷰가 먼저 열립니다. 생성된 아키텍처 결정과 전술 설계를 검토하고 「Plan 확정」을 누르면 「샌드박스 구현 열기」가 나타납니다.

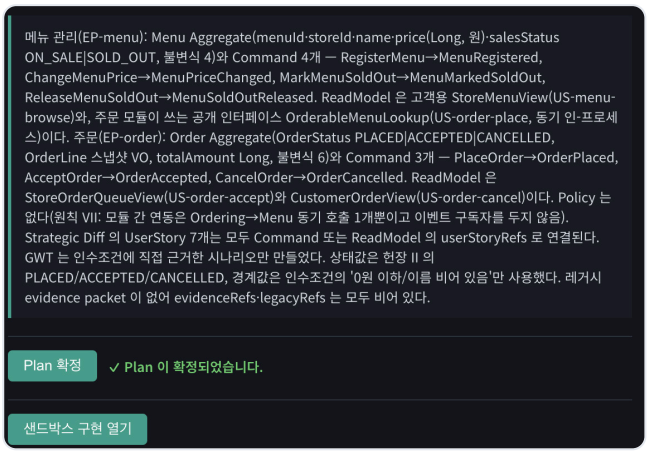


그림 17. Plan 요약과 확정

5단계 · 격리된 작업 공간에서 구현

「작업 목록 생성」으로 Phase별 작업을 만들고 「구현하기」를 누릅니다. 「Claude Code 셀로 이동」을 누르면 코드 탭에 프로포절 전용 셀이 열리고 구현이 시작됩니다. 셀에서 파일 작성 · 테스트 · 커밋 과정을 그대로 확인할 수 있습니다.

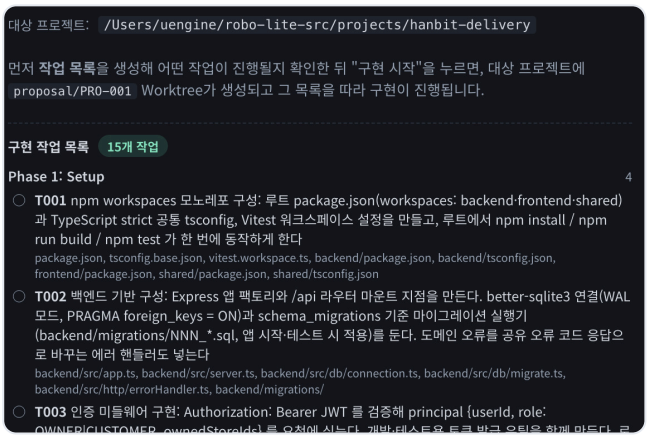


그림 18. 구현 작업 목록 15개 — 작업마다 생성할 파일 경로가 붙습니다.

6단계 · 검증과 승인

진행이 끝나면 「구현 완료 → 검증」을 누릅니다. 구조 검증과 인수 조건 결과를 확인한 뒤 「Accept (Dual Merge)」를 누르면 코드와 그래프 DB가 함께 갱신되고 상태가 ACCEPTED로 바뀝니다.

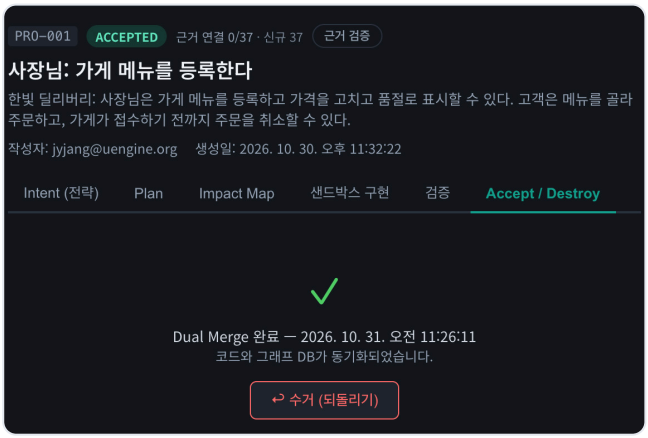


그림 19. Dual Merge 완료 — 「수거(되돌리기)」로 되돌릴 수 있습니다.

> **시간이 걸리는 단계는 해당 탭을 유지한 채 기다립니다.** Plan 생성 중에 그 탭을 새로고침하거나 다른 화면으로 이동하면 처음부터 다시 생성합니다. 기다리는 동안 다른 작업은 홈에서 새 탭을 열어 진행합니다.

13 · 보안

보안 · 거버넌스

Robo Architect Lite는 사용자 PC에서 실행하는 데스크톱 앱입니다. 도입 전에 **데이터 처리 범위와 변경 승인 절차**를 아래 기준으로 검토할 수 있습니다.

데이터 처리 범위

- **설계 데이터는 PC 안에 저장합니다.** 그래프 DB는 PC의 Docker 볼륨에 저장하며, 그래프 DB · Bolt · 분석 서비스 포트는 이 PC의 루프백 주소(127.0.0.1)에만 엽니다.
- **AI 요청은 Claude Code를 통해 처리합니다.** 프로포절의 분석 · 계획 · 구현은 PC의 Claude Code가 사용자의 로그인 세션으로 수행합니다. 요구사항과 설계, 코드 일부가 AI 요청에 포함되므로 조직의 Claude Code 사용 정책을 함께 검토해야 합니다.
- **레거시 분석의 AI 호출은 도구 없이 처리합니다.** 분석기의 요청은 Architect API를 거쳐 Claude Code가 답하며, 이때 도구 없이 빈 임시 폴더에서 텍스트만 생성합니다.
- **OpenAI API 키는 선택 사항입니다.** 문서 업로드와 설계 탭의 채팅 수정에만 사용합니다.
- **비밀 값은 로그에 남기지 않습니다.** 로컬 DB 비밀번호 같은 값은 사용자 계정 전용 파일이나 OS 암호화 저장소에 보관합니다.

변경 통제

항목	내용
계획 확인	최신 설계 원칙으로 확정한 Plan이 있어야 구현을 시작합니다
코드 격리	구현은 전용 브랜치의 워크트리에서만 수행하며, 대상 브랜치는 Accept 전까지 변경되지 않습니다
충돌 감지	같은 설계 요소를 다른 프로포절이 구현 중이면 Plan 단계 제출을 막고 알립니다
실패 항목 확인	검증에 FAIL이 있으면 실패 항목을 인지했다는 확인을 선택해야 Accept할 수 있습니다
동시 반영	그래프 반영이 실패하면 코드 병합을 되돌리고 실패 상태로 표시합니다
이력 · 로그	상태 변경마다 처리자와 시각을 기록하고, 요청마다 추적 번호를 붙여 로그를 남깁니다

AI 실행 방식 설정

프로포절 스킬은 기본으로 Claude Code CLI에서 실행하며, 환경 설정으로 OpenAI 호환 엔드포인트를 선택할 수 있습니다. 코드 탭의 구현 셀은 Claude Code를 사용합니다.

Lite는 개인용 앱으로 계정 · 권한 관리를 제공하지 않습니다. 작성자 정보는 PC의 Git 설정에서 읽습니다. 승인자 분리 같은 조직 정책은 도입 시 운영 절차로 정하고, 망 구성과 AI 사용 정책은 고객사 기준에 맞춰 협의합니다.

14 · 설치

설치 및 납품 방식

Robo Architect Lite는 데스크톱 앱 하나와 Docker 컨테이너 7개로 실행됩니다. 앱이 컨테이너 이미지를 적재하고 기동하므로 **Python과 데이터베이스를 따로 설치할 필요가 없습니다.**

구분	내용	제공 상태
Windows 10/11	설치 프로그램(약 100MB) · 컨테이너 이미지 묶음(약 1.5GB, 이미지 7개) · 해시 파일(SHA256SUMS). 사용자 단위로 설치하며 관리자 권한이 필요 없습니다	요청 시 제공
macOS	Apple Silicon용 데스크톱 앱(DMG)	설치 파일 배포 준비 중
첫 실행	이미지 묶음의 해시를 확인한 뒤 적재하고 컨테이너 7개를 기동합니다. 튜토리얼 PC에서는 약 50초, 설치본 첫 실행은 이미지 적재 때문에 4~5분이 걸렸습니다	—

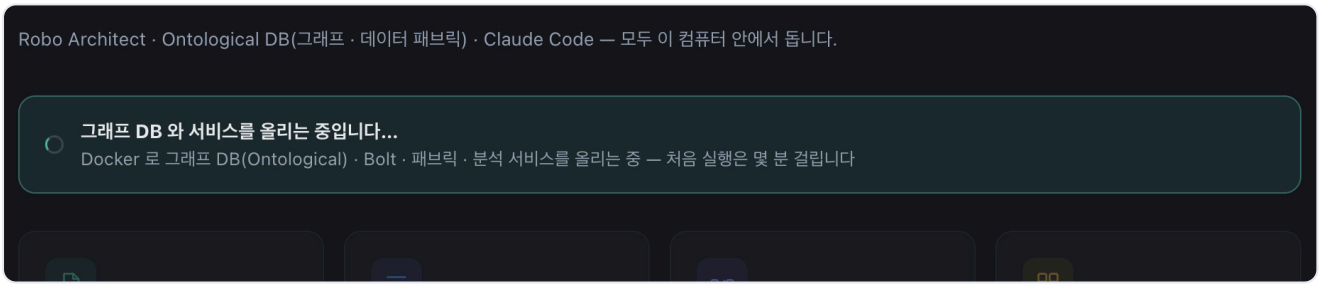


그림 20. 첫 실행 — 앱이 Docker로 그래프 DB와 분석 서비스를 올리는 동안 기능 카드가 잠겨 있습니다.

권장 실행 환경

PC

macOS 또는 Windows 10/11, 메모리 16GB 이상, 디스크 여유 공간 20GB 이상을 권장합니다.

Docker Desktop

그래프 DB와 분석 서비스가 컨테이너로 실행되므로 앱을 쓰는 동안 켜 두어야 합니다.

Claude Code

터미널에서 `claude` 를 실행해 로그인해 둡니다. 프로포절의 분석 · 계획 · 구현에 사용합니다.

Git

코드 폴더는 Git 저장소여야 하며, 프로포절마다 이 저장소의 워크트리를 만듭니다.

산출물은 고객 자산으로 남습니다. 구현 코드는 고객 Git 저장소에 작업별 커밋으로 남고, 설계 그래프는 PC의 그래프 DB에 저장되어 Bolt 프로토콜로 조회할 수 있습니다.

15 · 검증

제품 개발 및 품질 검증 체계

Robo Architect 개발팀은 **명세 작성부터 구현과 검증까지 일관된 절차**를 적용합니다. 기능마다 명세와 계획을 먼저 작성하고, 자동 시험과 실제 앱 실행으로 결과를 확인합니다.

명세를 먼저 작성하는 개발 절차

기능은 명세 작성, 구현 계획 수립, 작업 목록 작성, 구현 순서로 개발합니다. 산출물은 저장소에 함께 보관하며, 현재 **기능 폴더 59개에 기능 명세 58건, 구현 계획 42건, 작업 목록 39건**이 기록되어 있습니다.

자동 검증

85
백엔드 시험 파일
pytest · 시험 함수 725개

72
화면 시험 파일
Playwright

14
회귀 시험 파일
백엔드 서버 없이 실행

4
데스크톱 시험 파일
Electron 실행기 · 이미지 적재

회귀 시험(`npm run test:regression`)은 로직 단위 시험 2개와, 서버 응답을 미리 정한 데이터로 대신해 실행하는 화면 시험 12개로 구성합니다. 정적 분석은 ruff로 수행합니다.

실제 앱으로 확인한 사용 흐름

튜토리얼 14개 장의 화면과 결과는 macOS(Apple Silicon)에서 Lite 앱을 실제로 실행해 얻었습니다. 단계별 소요 시간은 다음과 같습니다.

단계	소요 시간	결과
INTENT(한 문장)	132초	프로세스 2 · 컨텍스트 2 · 스토리 7
Plan 생성	598초	Aggregate 2개 등 전술 설계 20개 항목
구현	약 22분	작업 15개 · 커밋 15건
검증 ~ 승인	약 7분	49건 중 PASS 44 · FAIL 0 · SKIP 5
레거시 분석	141초	클래스 6 · 메서드 18 · 테이블 7
변경 프로포절 PRO-002	80초 + 395초	INTENT + Plan, Order MODIFY · 충돌 가능성 HIGH
요구사항 문서 분석	171초	컨텍스트 5 · 기능 10 · 스토리 17

본 소개서의 화면은 모두 튜토리얼 실습 중 실제 제품에서 캡처했습니다. AI가 만드는 결과는 실행마다 조금씩 다릅니다. 같은 코드를 두 번 분석했을 때 업무 규칙은 35개와 53개였고, 클래스 · 메서드 · 테이블 수는 같았습니다.

16 · 레퍼런스

기반 제품 MSAEZ의 도입 실적

Robo Architect는 이벤트스토밍 기반 설계 · 코드 생성 도구인 **MSAEZ(MSA Easy)의 개발 및 운영 경험**을 바탕으로 확장한 제품입니다. 다음은 MSAEZ의 산업 현장, 공공 플랫폼, 교육 분야 활용 사례입니다.

대기업 실무
포스코DX
애플리케이션 · 마이크로서비스 설계와 소스 코드 생성 도구로 사용 중

공공 플랫폼
DPG 통합테스트베드
첨단 디지털 개발지원 도구(SaaS)로 등록·운영

9,655명
MSA School 누적 수강생
누적 교육 325회 · 17,864시간, 표준 실습 도구로 활용

① 산업 현장 — 포스코DX

MSAEZ는 포스코DX에서 **애플리케이션과 마이크로서비스의 설계 및 소스 코드 생성**에 사용되고 있습니다. 이벤트스토밍으로 도출한 설계 모델을 바탕으로 코드를 생성하는 데 활용됐습니다.

② 공공 — DPG 통합테스트베드

DPG(디지털플랫폼정부) 통합테스트베드는 한국지능정보사회진흥원(NIA)이 운영하는 플랫폼입니다. 중소기업, 스타트업, 시민 개발자, 공공기관이 민간 · 공공 데이터와 클라우드 자원을 활용해 **서비스를 개발 · 시험 · 검증**할 수 있도록 지원합니다. 민간의 SaaS를 개발 지원 도구로 제공합니다.

MSA Easy(MSAEZ)는 이 테스트베드에 등록되어 **마이크로서비스의 분석 · 설계 · 구현 · 배포 전 과정을 자동화**하는 SaaS로 제공되고 있습니다.

③ 교육 — MSA School

MSAEZ는 클라우드 네이티브 개발 교육기관 **MSA School의 표준 실습 도구**로 사용됩니다. 수강생은 이벤트스토밍으로 도메인을 설계하고, 모델에서 코드를 생성해 배포하는 과정을 실습합니다.

누적 교육 실적은 **325회, 수강생 9,655명, 17,864시간**입니다. 교육은 기초 · 중급 · 고급의 3단계로 구성되며, Biz(이벤트스토밍 · DDD), Dev(Spring Boot · Kafka), Ops(Kubernetes · GitOps)를 다룹니다.

구분	MSAEZ (기반 제품)	Robo Architect Lite
설계 · 구현	이벤트스토밍 설계, 모델에서 코드 생성	프로포절 단위 DDD 설계, Claude Code 구현 · 검증

초기 설계에서 지속적인 변경 관리로 확장했습니다. Robo Architect는 MSAEZ의 설계 · 코드 생성 기능을 바탕으로 변경 영향 분석, 프로포절 단계별 관리, 설계 원칙 기반 계획, 코딩 에이전트(Claude Code) 연동, 레거시 코드 근거 설계를 지원합니다.

※ 교육 수치는 MSA School 공개 집계 기준입니다. 구축 범위와 도입 효과는 영업 담당자가 개별 안내합니다.

16 · 도입

도입 절차 및 일정

환경 검토와 PoC(개념 검증)를 거쳐 본 도입까지 **4단계로 진행**합니다. 아래 일정은 표준 예시이며, 고객사의 환경과 검토 절차에 따라 조정합니다.



PoC 주요 검증 항목

- 실제 요구사항이 **바운디드 컨텍스트 · 사용자 스토리 · 인수 조건으로 적절히 구체화되는지**
- 기존 코드 분석 결과가 **새 프로포절의 설계 근거로 연결되는지**
- 기존 저장소 구성과 브랜치 운영 방식에 맞춰 **구현 · 병합을 진행할 수 있는지**
- 구조 검증과 인수 조건 결과로 **구현 결과를 판단할 수 있는지**
- Claude Code 사용과 데이터 처리 방식이 **고객사의 보안 정책을 준수하는지**

도입 시 협의할 사항

AI 실행 방식

Claude Code 사용 계정과 조직 정책, 데이터 처리 범위를 정합니다.

대상 저장소 · 브랜치 전략

작업할 저장소와 최종 병합 대상 브랜치를 정합니다.

프로젝트 설계 원칙

기술 스택, 아키텍처, 저장소 구성의 기준을 정합니다.

스킬 커스터마이징

조직의 설계 절차와 산출물 양식을 스킬에 반영합니다.

30분 제품 시연 및 PoC 환경 협의

요구사항 입력부터 설계 · 계획 · 구현 · 검증까지 실제 제품으로 시연합니다. 고객사의 업무 문서 일부를 미리 제공해 주시면 **해당 자료를 활용한 시연**을 준비합니다.

RA

Robo Architect Lite

AI ARCHITECTURE WORKBENCH

도입 문의

요구사항부터 구현까지, 변경을 함께 관리하세요.

도입 상담, 제품 시연, 견적을 문의해 주세요. 담당 전문가가 영업일 기준 1일 이내에 답변드립니다. 필요하시면 제품 시연 일정도 함께 안내합니다.

회사

(주)유엔진솔루션즈

제품

Robo Architect Lite

문서

Rev. 2026.10

프로포절 하나로 이어지는 네 단계

요구사항
문장 · 문서에서 DDD 설계로

계획
설계 원칙에 근거한 결정

구현
격리된 작업 공간 · Claude Code

검증 · 반영
코드 · 설계 그래프 동시 반영

(주)유엔진솔루션즈 · uEngine Solutions, Inc.

© 2026 uEngine Solutions. 본 문서 Rev. 2026.10 · 기재된 수치 · 화면은 작성 시점 기준이며 사전 통지 없이 변경될 수 있습니다.

uEngine