객체지향개발방법론
팀프로젝트#7

2026.06.11

7팀 강희진 김희용 양지윤 윤성진

RT-DevOps OOAD
: Risk-Tiered, Agent-Paired OOAD on DevOps

사람 ↔ AI

위험도에 따라
주도권 분담

Dev ↔ Ops

공통 파이프라인
으로 연결

속도 ↔ 품질

초기 가속 +
결정 게이트

"Agent는 SSD, 즉 개발 초기부터 함께 제안하고, 사람은 위험도에 따라 결정한다."

독자적 개발방법론의 근거

프로세스 모델

단계마다 입력·산출물·
완료 게이트 정의

역할 정의

사람·Agent 책임 경계 명시 (ROLES.md)

원칙

6대 불변 원칙이 가치관 고정 (CLAUDE.md)

산출물 명세

UC·SSD·Domain·SD/CD·
Code·Test 전부 정의

활동과 실천

13개 슬래시 커맨드로 활동 코드화

반복 적용성

방법론 고정, 프로젝트 정보 4개 파일만 교체

→ 단순한 "AI 활용 가이드"가 아니라, 6가지 요건을 모두 갖춘 독자적 개발방법론입니다.

방법론의 재료

UP / OOAD

- 단계 골격
  - Inception → OOA → OOD → OOI → Ops
- 동적 모델 주도
  - 추적성
  - maintenance nightmare 방지

DevOps

- Dev ↔ Ops 상충 해소
- 공통 CI/CD 파이프라인
  - Ops 피드백 루프

Oxford Martin AIGI

Risk Tiers: Towards a Gold Standard for Advanced AI

- 핵심 차별 출처
- 예상 피해 기준 등급화 "완화책"의 자리에 사람-AI 분담" 삽입 → 도메인 전이

사람-AI 역할 분담의 철학

"AI가 있다면, 사람은 무엇을 해야 하는가?"

고위험 트랙

- 사람이 SD·CD를 직접 설계
- Agent가 SOLID 위반 검증
- 사람이 먼저 명세로 테스트 작성
- Agent가 테스트를 통과하는 코드 생성

→ **명세 우선 TDD**

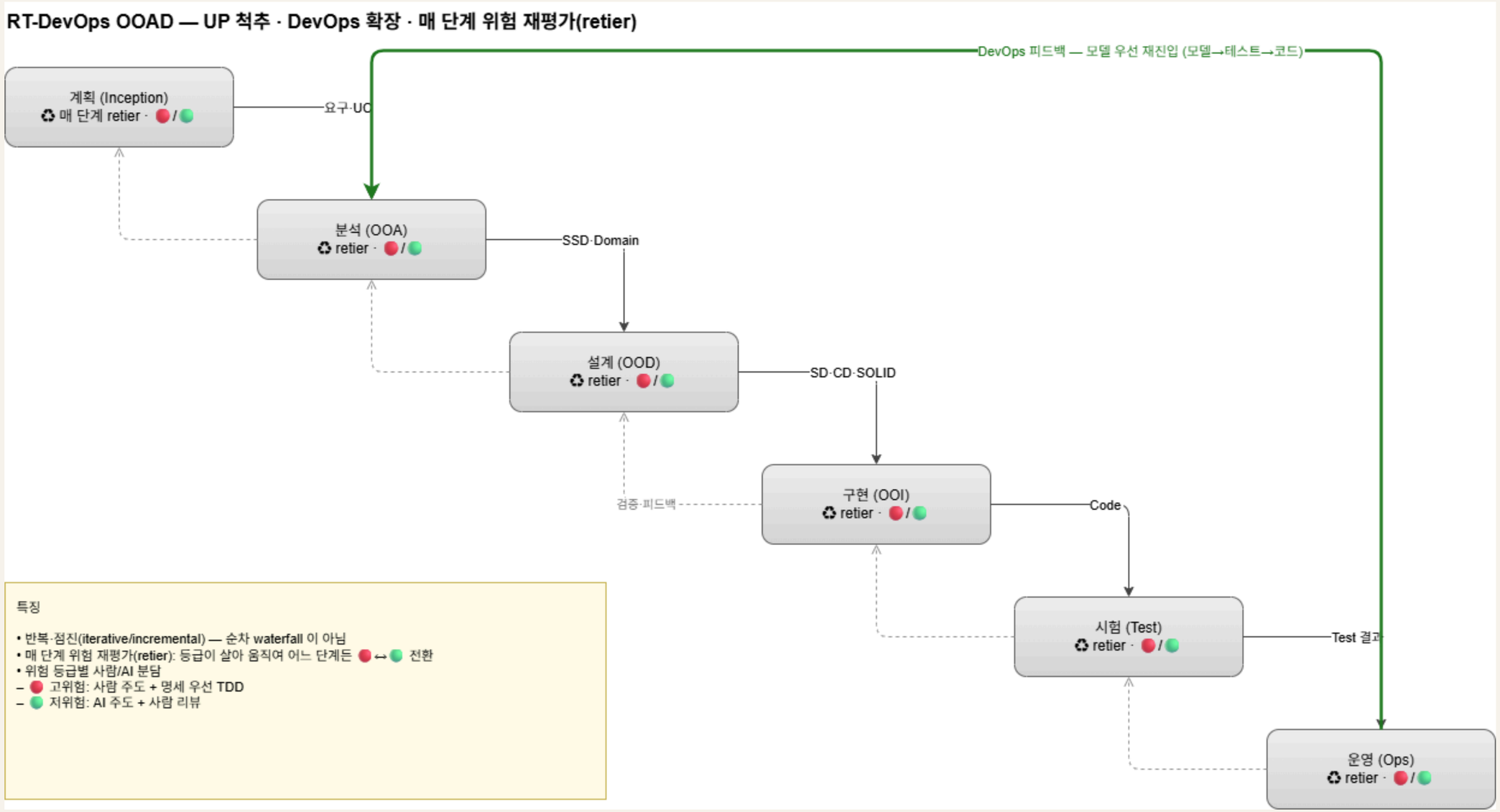
VS

저위험 트랙

- Agent가 설계·구현 주도
- 사람이 결과물 리뷰 및 승인
- Agent vibe coding으로 빠른 구현
- 공통 CI/CD 게이트 통과 필수

"Agent proposes, Human decides." — 어떤 단계에서도, 어떤 등급에서도 변하지 않는 원칙

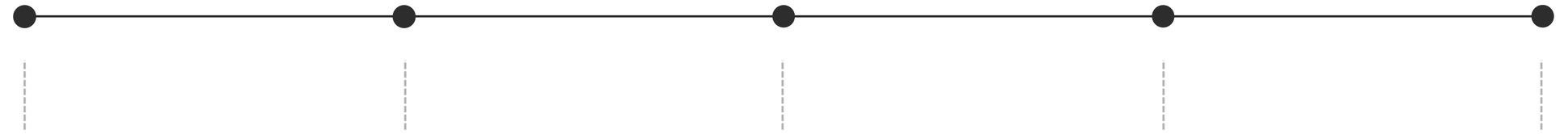
전체 개발 흐름


RT-DevOps OOAD — UP 척추 · DevOps 확장 · 매 단계 위험 재평가(retier)
DevOps 피드백 — 모델 우선 재진입 (모델→테스트→코드)
계획 (Inception)
♻ 매 단계 retier · 🔴/🟢
요구·UC
분석 (OOA)
♻ retier · 🔴/🟢
SSD·Domain
설계 (OOD)
♻ retier · 🔴/🟢
SD·CD·SOLID
구현 (OOI)
♻ retier · 🔴/🟢
검증·피드백
Code
시험 (Test)
♻ retier · 🔴/🟢
Test 결과
운영 (Ops)
♻ retier · 🔴/🟢
특징
• 반복·점진(iterative/incremental) — 순차 waterfall 이 아님
• 매 단계 위험 재평가(retier): 등급이 살아 움직여 어느 단계든 🔴↔🟢 전환
• 위험 등급별 사람/AI 분담
– 🔴 고위험: 사람 주도 + 명세 우선 TDD
– 🟢 저위험: AI 주도 + 사람 리뷰

절차

1. **Pre-Inception** : /sketch · /srs
   - 문제: 요구가 메모·대화 수준, 누락 파악 어려움
   - 스케치 → Agent가 요구 후보 추출 → SRS.md 확정
2. **Inception** : /inception
   - 문제: 위험 미파악, 계획·아키텍처 부재
   - 첫 위험 등급 분류 → 고/저위험 트랙 배정
3. **OOA** : /ssd · /domain
   - 문제: 내부 설계 전에 외부 경계 미확정
   - SSD로 블랙박스 경계 확정 → Domain Model
4. **OOD** : /ood
   - 문제: 코드와 모델 불일치, 추적성 부재
   - SDD.md 확정 → 모든 코드의 진실 소스
5. **OOI** : /code · /test · /review
   - 문제: 환각·명세 이탈·품질 미확보
   - 위험 트랙별 TDD·vibe coding + CI/CD 게이트
6. **Ops** : /ops · /retier
   - 문제: 코드 먼저 고치면 모델 붕괴
   - 모델·테스트 우선 수정 → cascade-stale

산출물 사슬 & Cascade-Stale



Cascade-Stale

- 상위 단계에 변경사항이 있으면 하위 단계도 자동 stale 처리됨
  - ex) SSD를 고치면 SD, CD, Code, Test 전부 재실행
- 잘못된 내용이 하위 단계로 전파되지 않음

불변 버전 기록

- 모든 산출물은 records 디렉토리에 버전으로 관리됨
  - --rollback : 어떤 시점으로도 복귀 가능
  - --alt : 대안을 여러개 생성해 병렬로 비교
- 설계 결정의 이유가 모두 기록됨

반복 구조

① UP 거시 반복

UC 단위: Inception → OOA → OOD → OOI → Ops
매 이터레이션마다 /retier 위험 재평가 등급이 바뀌면 트랙도 따라 바뀐다

② Proposed → Accept 검증 루프

Agent 초안(proposed) → 사람 검토 → 재시도(--retry) or 승인(--accept)
SSD, SD, Code, Review 어느 단계에서나 동일하게 작동 품질은 매 산출물마다 즉시 확인

③ Ops → Dev 피드백 루프

AI 모니터링 → 이슈·PR 초안 → 모델 우선 수정 → CI/CD 게이트
배포 이후에도 방법론이 살아있음 DevOps 순환 완성, 시스템이 운영 중에도 진화

기존 방법론과의 비교

비교 대상	한계	RT-DevOps OOAD의 개선점
순수 UP/RUP	AI 협업 개념 없음, 요구가 이미 있다고 가정	**요구 구체화 단계부터 Agent 활용, 위험도 기반 동적 역할 배분**
Scrum/Agile	위험 차등 약함, 설계 경시, 추적성 부재	**위험도 차등 분류, AI 입력 규정, 기록 기반 추적**
AI Vibe Coding만	환각, 명세 이탈, 유지보수 붕괴	**AI 입력 규정, 모델 기반 수정 수행**

감사합니다.