AI · OOAD · DevOps

Optimized and Balanced SW Development Process

3팀
202213351 김태성
202111382 최성준
202011434 최원탁
202011380 최용근

CONTENTS

목차

01 OVERVIEW

방법론 개요

AI-Balanced UP DevOps Process는 객체지향 분석·설계 중심의 UP/OOAD 방법론에 AI Coding Agent와 DevOps 운영 루프를 결합한 개발 프로세스입니다.

1 반복적·점진적 개발

UP 기반으로 요구사항·설계·구현·배포를 단계적으로 안정화

2 OOAD 산출물 기준

Use Case, Domain/Class/Sequence Diagram을 개발 기준 문서로 사용

3 AI Coding Agent

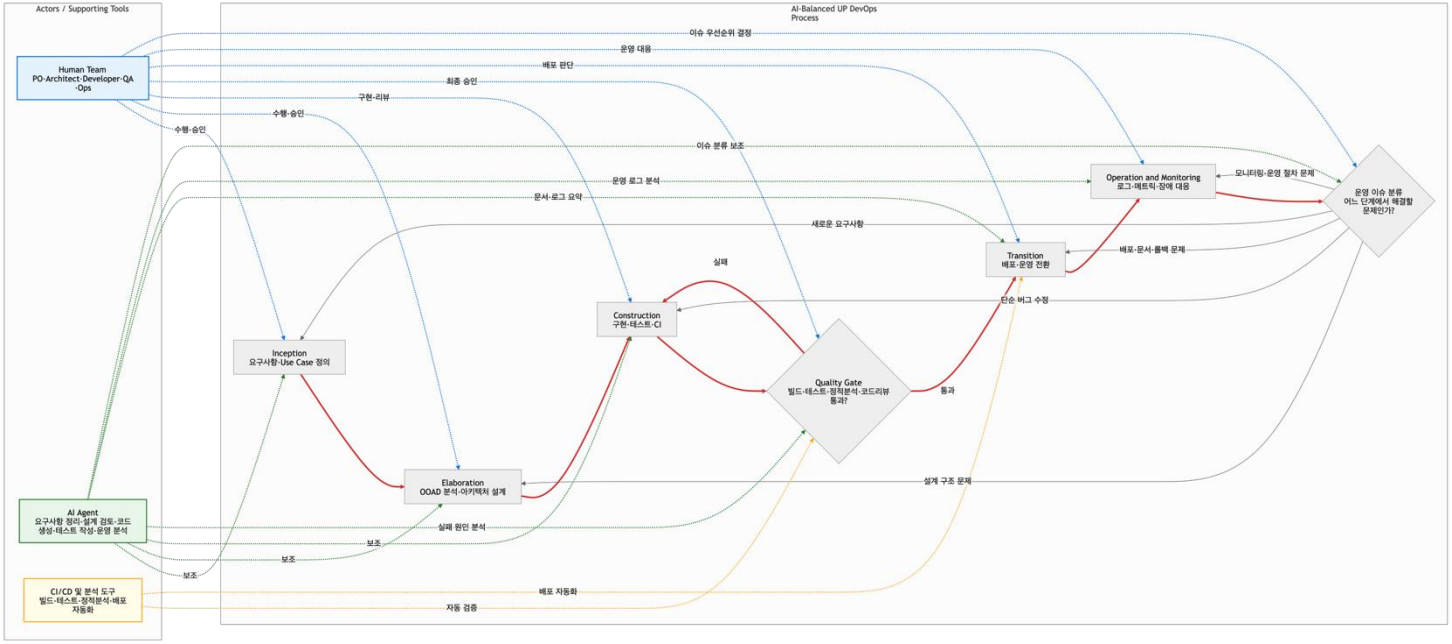
요구사항 정리·설계 검토·코드 생성·테스트·CI 분석·로그 요약 보조

4 DevOps 운영 루프

배포 이후 운영 데이터·장애 분석이 다시 요구사항과 설계로 환류

전체 그림

UP 5단계 흐름과 AI Agent · Human Team · Quality Gate의 지원 관계



프로세스 단계

Inception → Elaboration → Construction → Transition → Operation

참여 주체

AI Agent · Human Team이 전 단계 지원

품질 게이트

Construction과 Transition 사이 검증 · 통과/실패 분기

방법론의 핵심 구조

UP 4단계 + Operation & Maintenance 단계 추가

단계	목적	주요 산출물	AI Agent 활용
Inception	요구사항·범위 정의	Vision, Use Case, Actor	요구사항 분류, UC 초안
Elaboration	도메인 분석·아키텍처	Domain/Class/Sequence	클래스 후보, 설계 검토
Construction	기능 구현·테스트	코드, 테스트, PR, CI	코드 생성, 테스트, 리팩터링
Transition	배포·운영 전환	Release Note, Runbook	배포 로그 분석, 문서 초안
Operation & Maintenance	유지보수·피드백	운영 로그, 장애 리포트	로그 요약, 원인 분석

역할 정의

AI Agent는 개발자를 대체하지 않는다 — 초안 작성자, 구현 보조자, 검증 도우미

역할	담당 범위
PO(Product Owner)	요구사항 우선순위, Use Case 승인, 비즈니스 규칙 확정
OOAD Architect	도메인 모델, 클래스 책임, 아키텍처, 인터페이스 결정
Developer	AI 생성 코드 검토, 직접 구현, 리팩터링, PR 작성
QA(Quality Assurance)	테스트 전략, Acceptance Test, 회귀 테스트 기준
DevOps / Ops	CI/CD, 배포, 모니터링, 장애 대응, 운영 피드백
AI Coding Agent	문서 초안, 설계 대안 제안, 코드·테스트 생성, 로그·CI 분석

04.1 AI AGENT SCOPE

AI Agent의 허용 범위와 제한 범위

✓ 허용 범위

- 요구사항 요약
- Use Case 초안 작성
- 클래스 후보 제안
- 코드 생성 및 수정
- 테스트 코드 생성
- CI 오류 원인 분석
- 운영 로그 요약
- 문서 업데이트 제안

✗ 제한 범위

- 요구사항 최종 확정
- 비즈니스 우선순위 결정
- 최종 아키텍처 결정
- main 브랜치 직접 merge
- 실패한 테스트 무시
- 품질 기준 임의 완화
- 운영 장애 최종 판단
- 보안 정보·운영 secret 접근

05.1 STAGE 1

Inception — 요구사항 정의

시스템의 범위와 핵심 Use Case를 명확히 한다

수행 절차

1. PO가 사용자 요구사항·문제·목표 기능 작성
2. AI가 기능·비기능·예외 상황으로 분리
3. PO·Architect 검토 후 범위 확정
4. AI가 Use Case·Actor 후보 제안
5. 팀이 핵심 Use Case 선택·우선순위 결정

주요 산출물

- Vision 문서
- Actor 목록
- Use Case Diagram·명세서
- 비기능 요구사항

완료 기준

- ✓ 핵심 Actor 정의
- ✓ 주요 Use Case 식별
- ✓ 기본·예외 흐름 작성
- ✓ 기능 범위 명확화
- ✓ 운영 요구사항 1개 이상 포함

05.2 STAGE 2

Elaboration — OOAD 분석 및 설계

구현 전에 객체지향 구조와 아키텍처 위험을 줄인다

수행 절차

1. Architect가 Use Case 기반 도메인 개념 식별
2. AI가 Entity·Value Object·Service 후보 제안
3. Architect가 Domain Model 확정
4. Use Case별 Sequence Diagram 작성
5. AI가 책임 과부하·순환 의존성 검토
6. DevOps가 CI/CD·모니터링 설계

주요 산출물

- Domain Model
- Class·Sequence Diagram
- Component Diagram
- API/Interface 명세
- 테스트 전략, CI/CD 설계

완료 기준

- ✓ 핵심 UC별 Sequence Diagram
- ✓ Domain Class 책임 정의
- ✓ 아키텍처 구조 확정
- ✓ 테스트 전략 수립
- ✓ CI/CD 품질 게이트 기준
- ✓ 운영 로그·모니터링 항목 정의

05.3 STAGE 3

Construction — 구현 및 테스트

Use Case 단위의 작은 반복 개발로 진행한다

수행 절차

1. 구현할 UC·Acceptance Criteria 선택
2. AI에 UC 명세·다이어그램·기존 코드 제공
3. AI가 구현 계획 먼저 작성
4. Developer가 계획 검토·범위 제한
5. feature branch에서 코드·테스트 생성
6. Developer 검토 후 PR 생성
7. CI 실패 시 AI가 로그 분석·수정안 제안
8. 품질 게이트 통과 후 merge

주요 산출물

- Production Code
- Unit·System Test
- PR 설명, 리뷰 체크리스트
- CI 실패 분석

완료 기준

- ✓ 빌드 성공
- ✓ Unit·System Test 통과
- ✓ 정적 분석 통과
- ✓ 코드 리뷰 승인
- ✓ 문서·코드 일치
- ✓ 로그·예외 처리 운영 기준 충족

05.4 STAGE 4

Transition — 배포 및 운영 전환

개발된 기능을 사용자/운영 환경에 안정적으로 전달한다

수행 절차

1. DevOps가 staging 환경에 배포
2. QA가 Acceptance Test 수행
3. AI가 테스트·배포 로그 요약
4. Developer가 문제 수정
5. 체크리스트·롤백·대시보드 확인
6. 승인 후 production 배포
7. 로그·에러율·응답시간·피드백 수집

주요 산출물

- Release Note
- Deployment Checklist
- Rollback Plan
- 운영 Runbook
- Monitoring Dashboard 기준

완료 기준

- ✓ 배포 성공
- ✓ 핵심 기능 Acceptance Test 통과
- ✓ 롤백 절차 준비
- ✓ 운영 로그·지표 확인 가능
- ✓ Release Note 기록

05.5 STAGE 5

Operation & Maintenance — 운영 및 유지보수

DevOps 관점에서 소프트웨어 개발은 배포 후에도 계속된다

운영 이슈별 이동 단계

운영 중 발견된 문제	이동 단계
특정 API에서 NullPointerException	Construction
도메인 책임이 Service에 과도하게 몰림	Elaboration
사용자가 새로운 권한 기능 요구	Inception
배포 후 응답 시간이 느림	Elaboration 또는 Construction
장애 대응 문서 부족	Transition 또는 Operation

핵심 규칙

6.1 AI는 초안·보조 역할

최종 결정은 사람 — 요구사항·아키텍처·도메인 모델·DB 스키마·배포·main merge는 사람이 승인

6.2 Use Case 단위 개발

(X) "회원 시스템 전체 구현해줘" → (O) "이메일·비밀번호 로그인 UC, JWT 반환, 기존 구조 유지로 계획 먼저"

6.3 OOAD 문서·코드 동기화

PR에 코드 변경 · 테스트 · 관련 다이어그램 · 운영 영향 · 롤백 필요 여부를 함께 포함

6.4 완료 기준에 운영 조건 포함

"동작한다"로 끝내지 않음 — 테스트·리뷰·예외·로그·모니터링·배포·롤백·문서까지 충족

제3자가 따라할 수 있는 적용 절차

1. 프로젝트의 핵심 문제·목표를 한 문단으로 작성
2. AI에게 요구사항을 기능·비기능·예외로 분류 요청
3. PO·Architect가 결과 검토 후 범위 확정
4. AI에게 Actor·Use Case 후보 생성 요청
5. 팀이 핵심 UC 선택·우선순위 결정
6. Architect가 Domain·Class Diagram 작성
7. AI에게 책임·의존성·순환 참조 검토 요청
8. Developer가 UC 하나 선택
9. AI에게 구현 계획 먼저 작성 요청
10. Developer가 계획 검토·범위 제한
11. AI가 코드·테스트 초안 작성
12. Developer가 품질·설계·예외 처리 검토
13. CI에서 빌드·테스트·정적 분석 실행
14. 실패 시 AI에게 로그 분석·수정 후보 요청
15. 품질 게이트 통과 후 PR 생성
16. Reviewer가 코드·OOAD 문서 일치 확인
17. Staging 배포·Acceptance Test 수행
18. Production 배포 후 지표·피드백 수집
19. 운영 이슈를 버그·설계·신규 요구사항으로 분류
20. 분류에 따라 Construction·Elaboration·Inception에 재반영

최종 정리

AI-Balanced UP DevOps Process

UP/OOAD의 체계적인 분석·설계 절차를 유지하면서, AI Coding Agent를 전 단계에 활용하는 방법론

1

개발 속도

AI Coding Agent

각 단계의 초안·생성·분석을 AI가 보조하여 속도를 끌어올린다

2

설계 품질

OOAD

Use Case·Domain·Sequence 기반으로 객체지향 설계 품질을 유지한다

3

운영 안정성

DevOps

배포 후 운영 피드백을 다시 요구사항·설계에 반영한다

Thank You

감사합니다

Optimized and Balanced SW Development Process