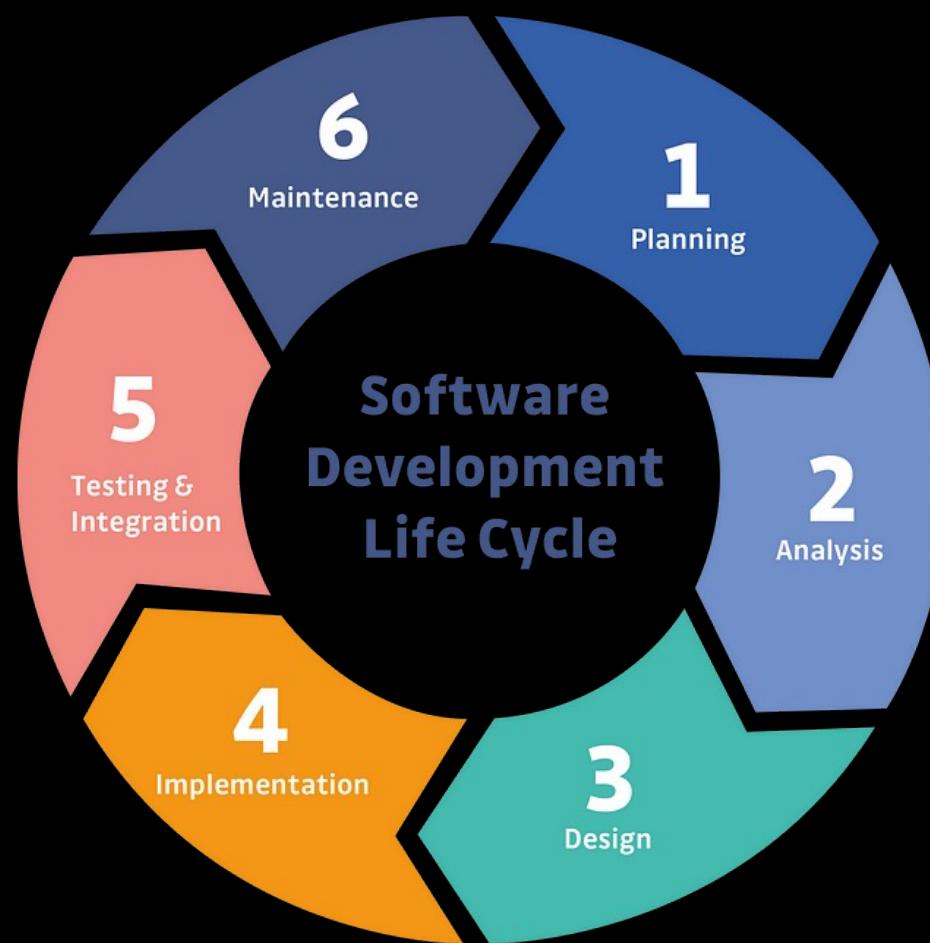


SDLC 정의 및 주요 단계



SDLC란 무엇인가?

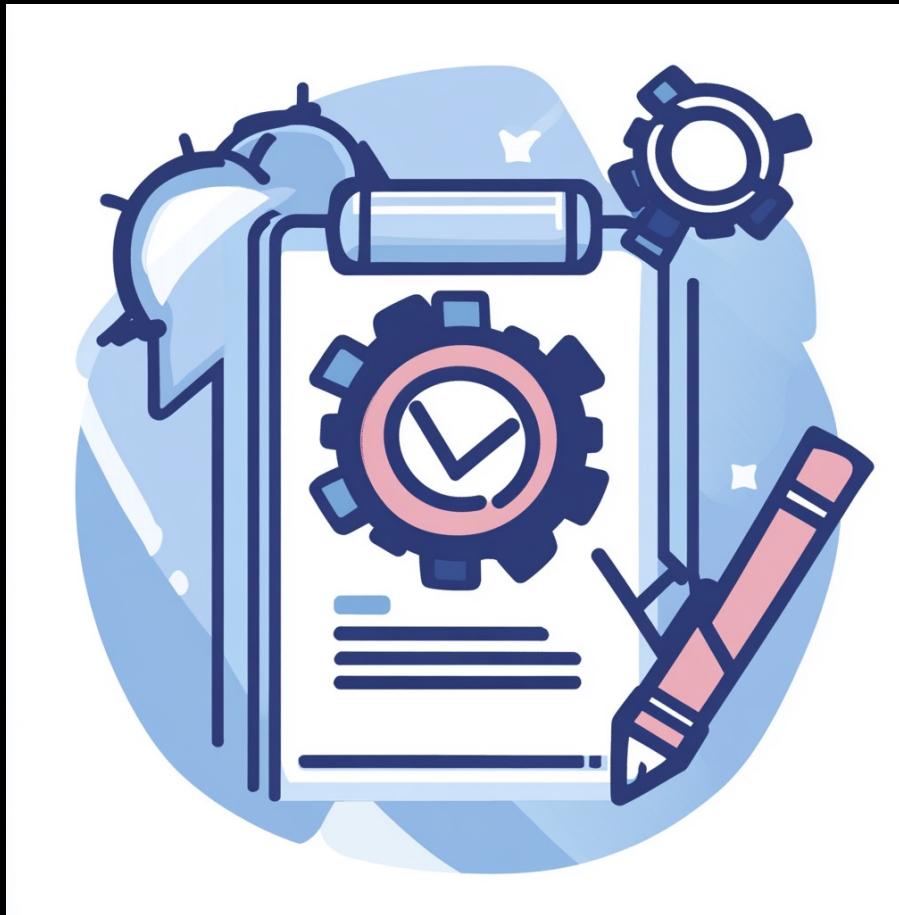


Software Development Life Cycle

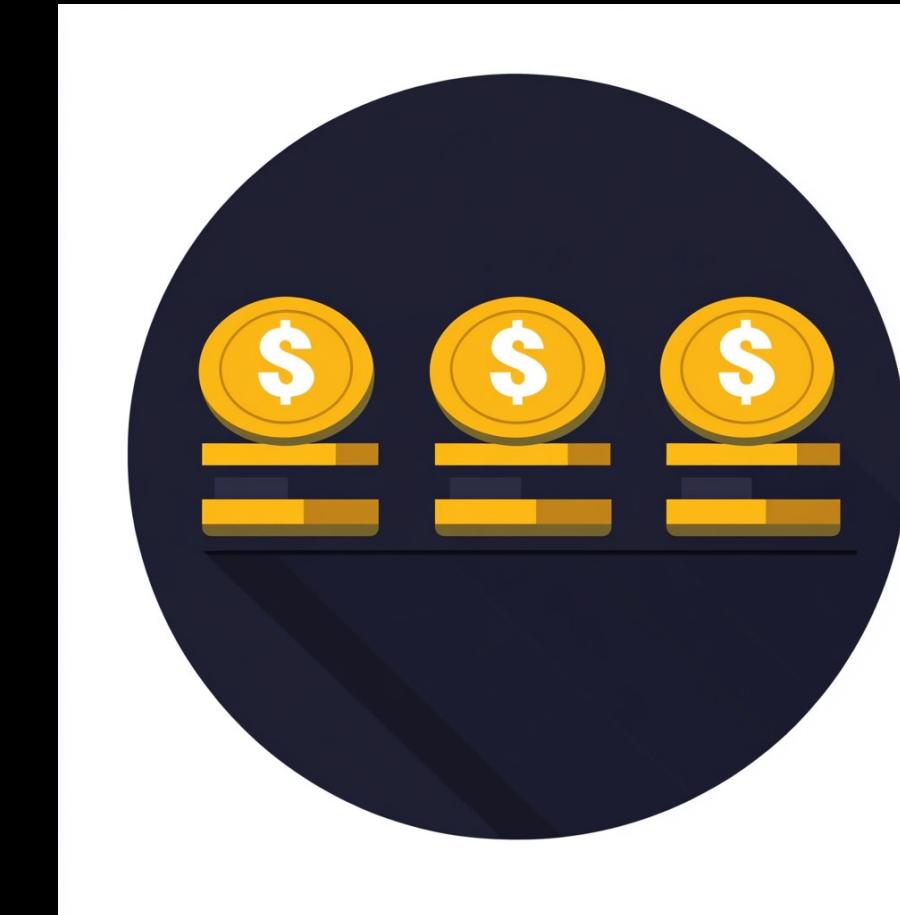
- 소프트웨어 개발을 체계적으로 관리하는 과정
- 계획, 요구사항 정의, 설계, 구현, 테스트, 배포, 유지보수로 구성



SDLC의 중요성



퀄리티 개선



비용 절감

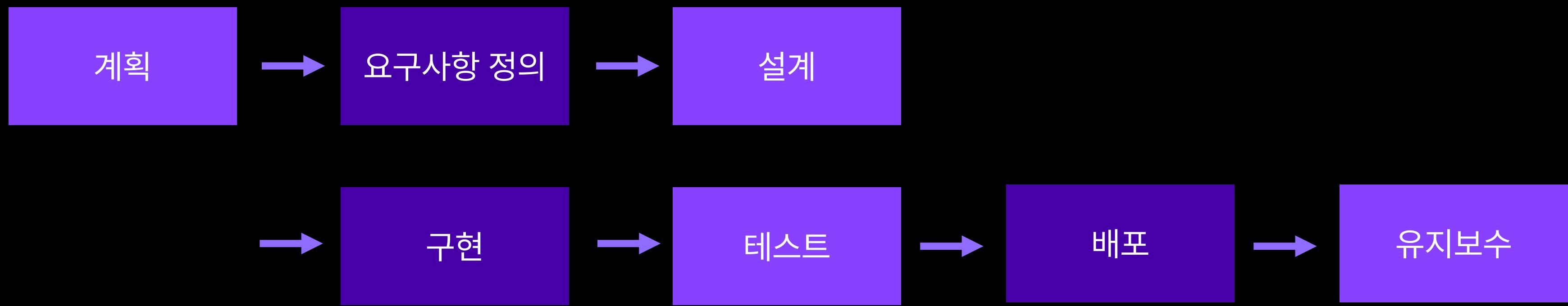


팀 간 협업



SDLC와 품질 보증(QA)의 관계

[Software Development Life Cycle(SDLC)]



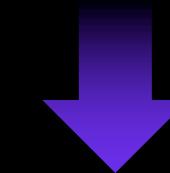
품질 보증(QA) 단계 수행



소프트웨어 품질 보장의 중요성



초기 단계에서 결함 예방 및 비용 절감



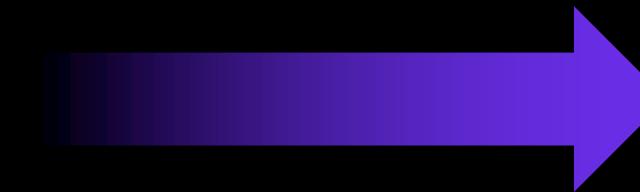
높은 품질의 소프트웨어



높은 고객 만족도 및 장기적인 성공 보장



SDLC와 프로젝트 성공



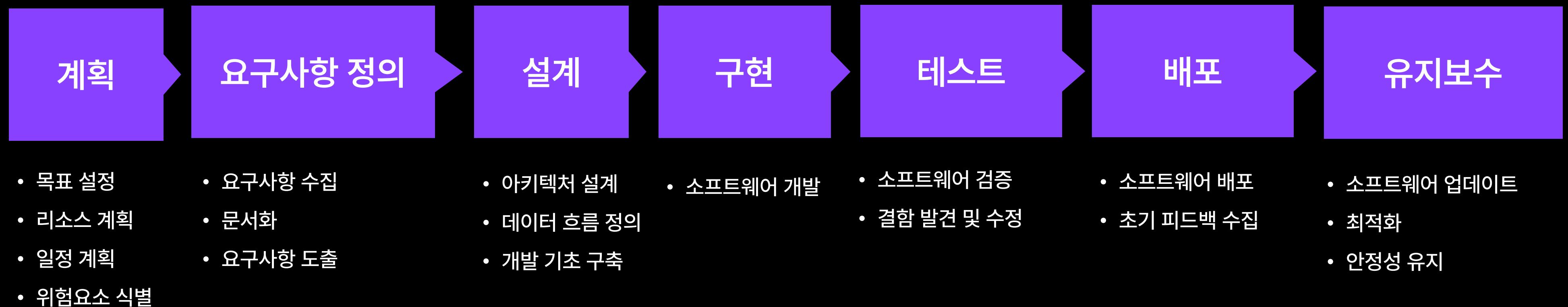
일정 관리
퀄리티 향상



성공적인 프로젝트

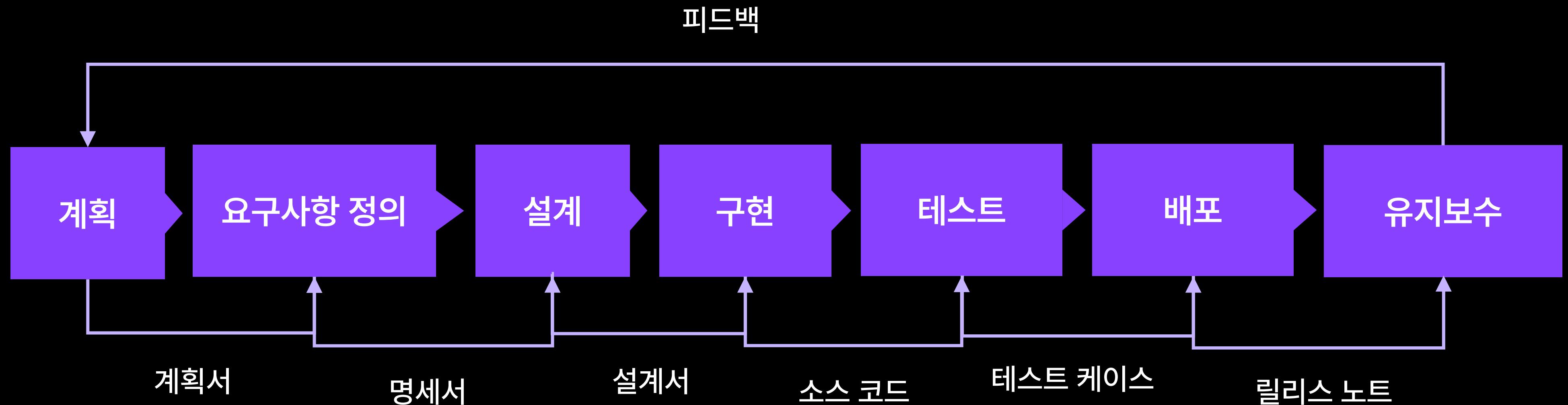


SDLC 주요 단계



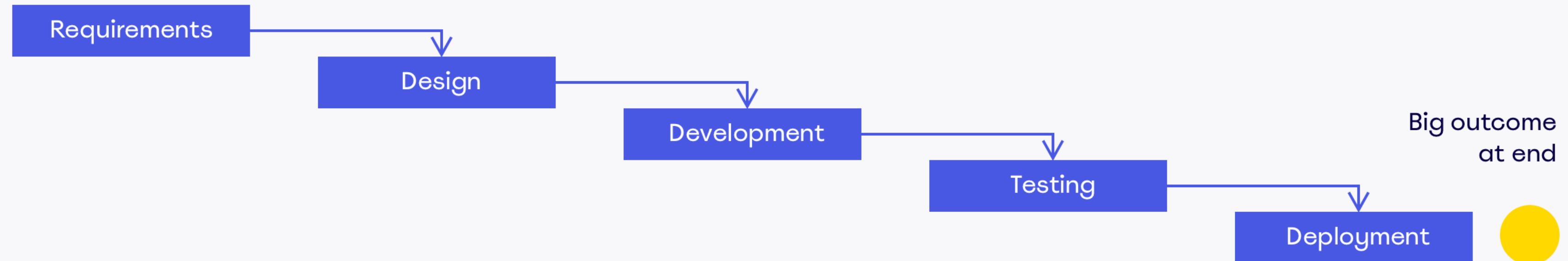


SDLC 단계 간 상호작용

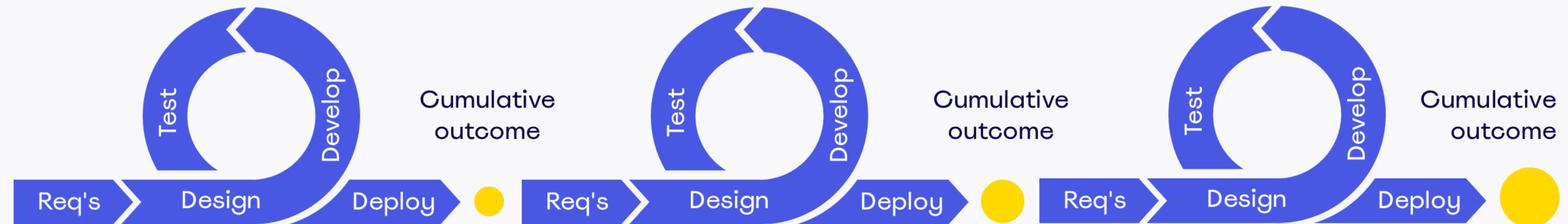




Waterfall



Agile





SDLC 모델

특징	Waterfall 모델	Agile 모델
프로세스 흐름	순차적 단계 진행	반복적, 점진적 개발
변화 수용	어려움	용이
고객 피드백 반영	최종 제품 단계에서 반영	각 스프린트마다 반영 가능
적용 사례	요구사항 고정 프로젝트	요구사항이 자주 변경되는 프로젝트
장점	명확한 계획과 예측 가능성	유연성, 빠른 피드백
단점	변화에 비효율적, 긴 개발 주기	관리 복잡성, 일정 예측 어려움



SDLC 미적용 시 문제점



비용 ↑

시간 ↑

품질 ↓

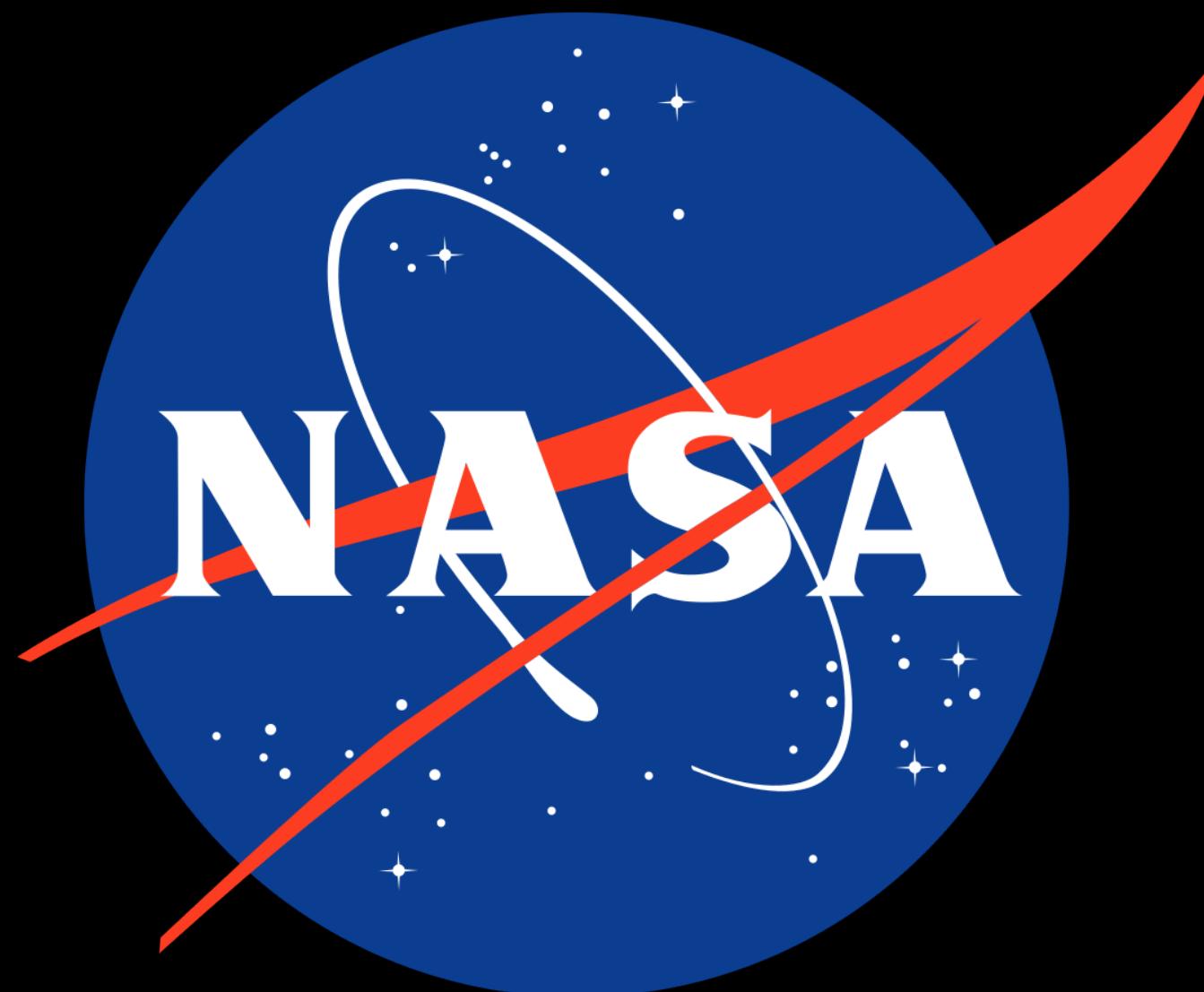
고객 만족도 ↓



성공적인 SDLC 적용 사례 (1)



NASA – 우주선 소프트웨어 개발



- SDLC 적용으로 우주 탐사 소프트웨어의 품질과 안정성 확보
- 요구사항 정의 단계 - 모든 기능 문서화
- 설계 단계 - 엄격한 데이터 흐름 관리
- 테스트 단계 - 다양한 시뮬레이션 실행



성공적인 SDLC 적용 사례 (2)

아마존 – 전자상거래 플랫폼 개발

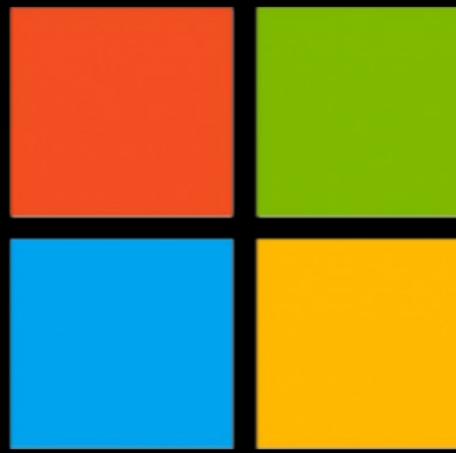


- SDLC +Agile 모델 결합
- 요구사항 정의 단계 – 고객 경험 데이터 활용
- 지속적 통합 및 배포



성공적인 SDLC 적용 사례 (3)

마이크로소프트 – Windows 운영 체제 개발



Microsoft

- SDLC 원칙 기반으로 개발 프로세스 관리
- 설계 단계 – 모듈화된 아키텍처 구축
- 테스트 단계 – 수백만 건의 오류 데이터 분석
- 운영체제 품질 유지 및 사용자 경험 개선



SDLC의 주요 목적



프로젝트 성공



품질 보장



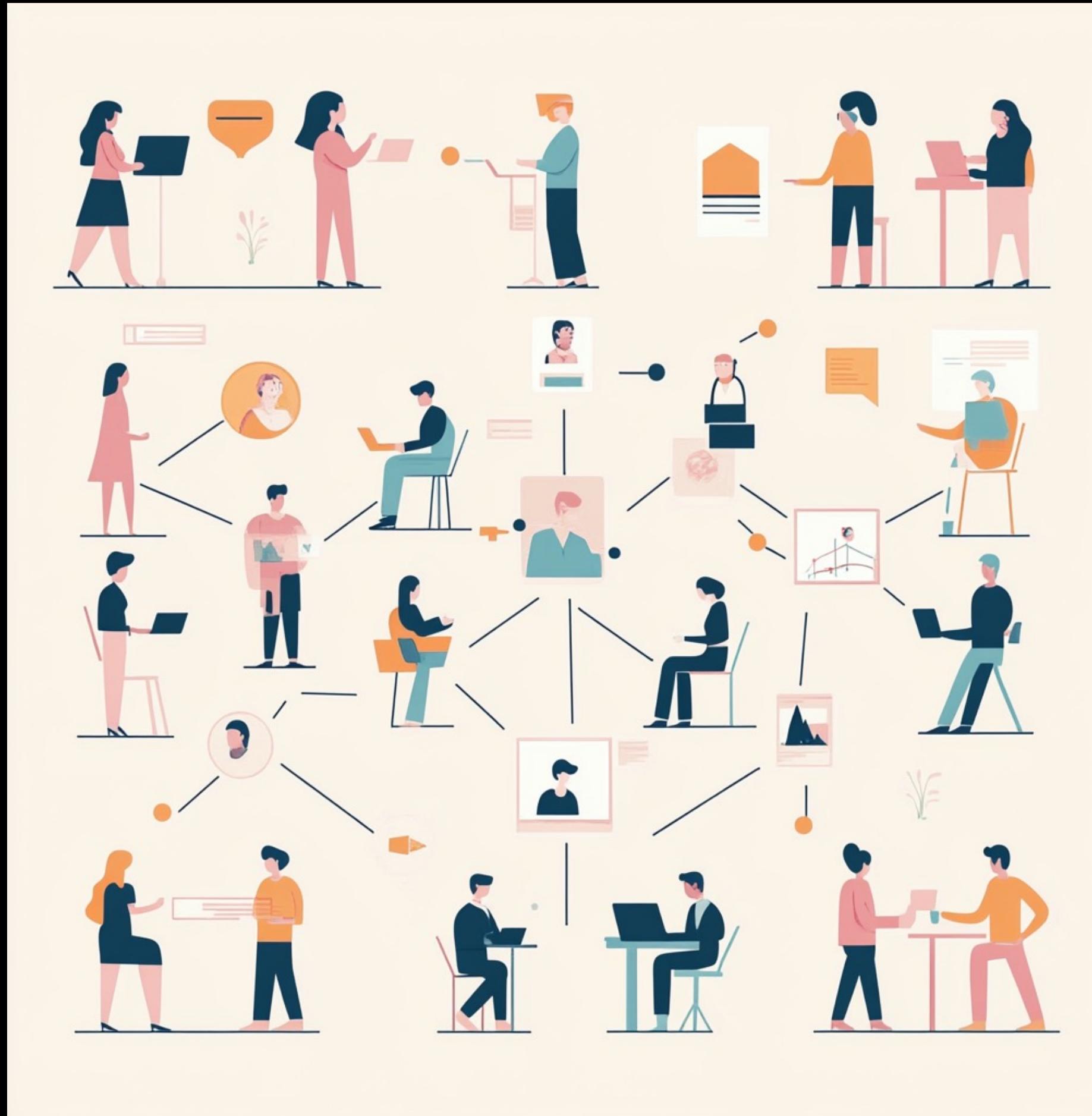
효율성 증대



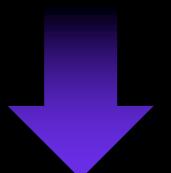
리스크 관리



SDLC의 필요성



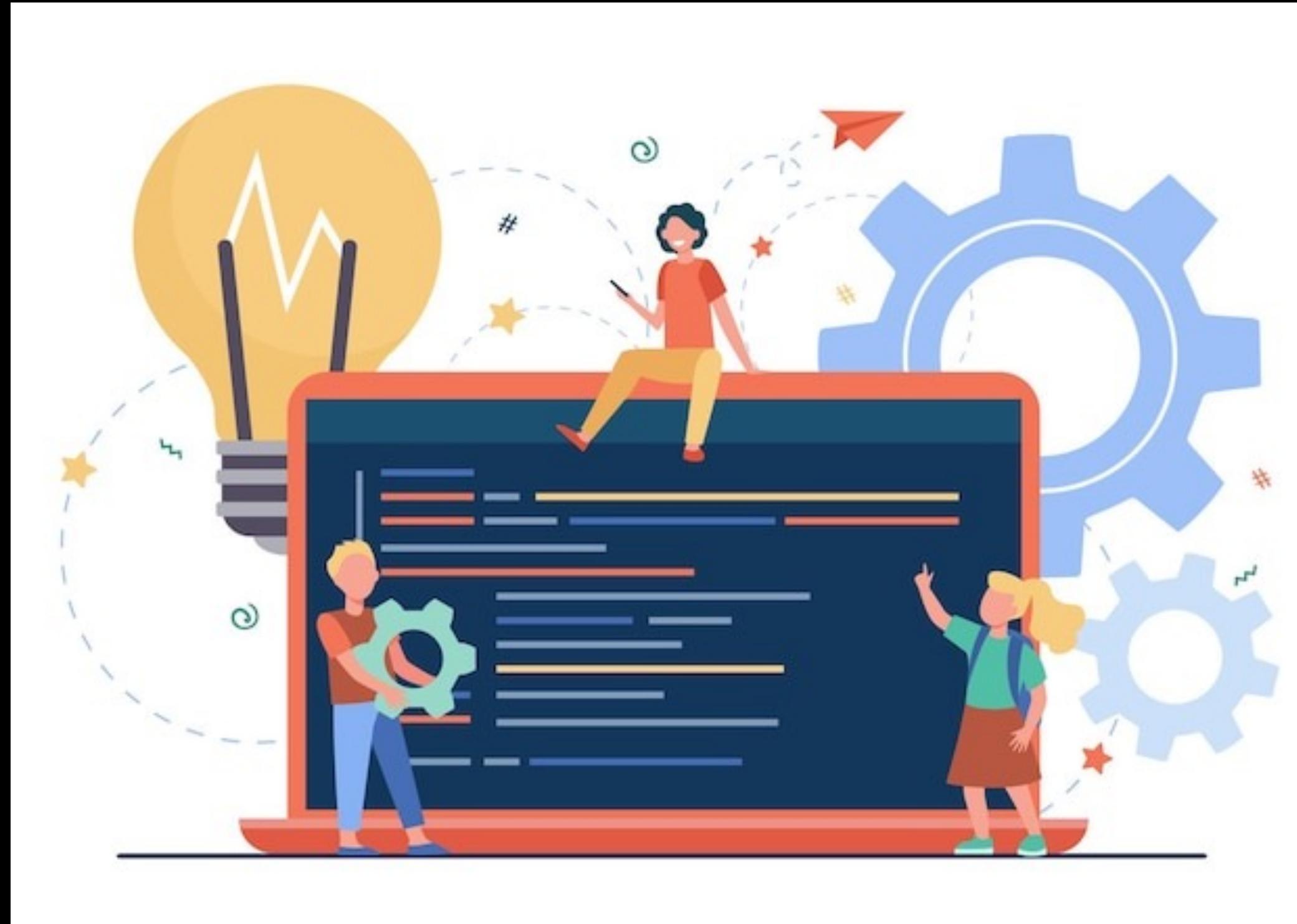
명확한 단계별 목표 설정



팀 간 협업 촉진, 원활한 의사소통



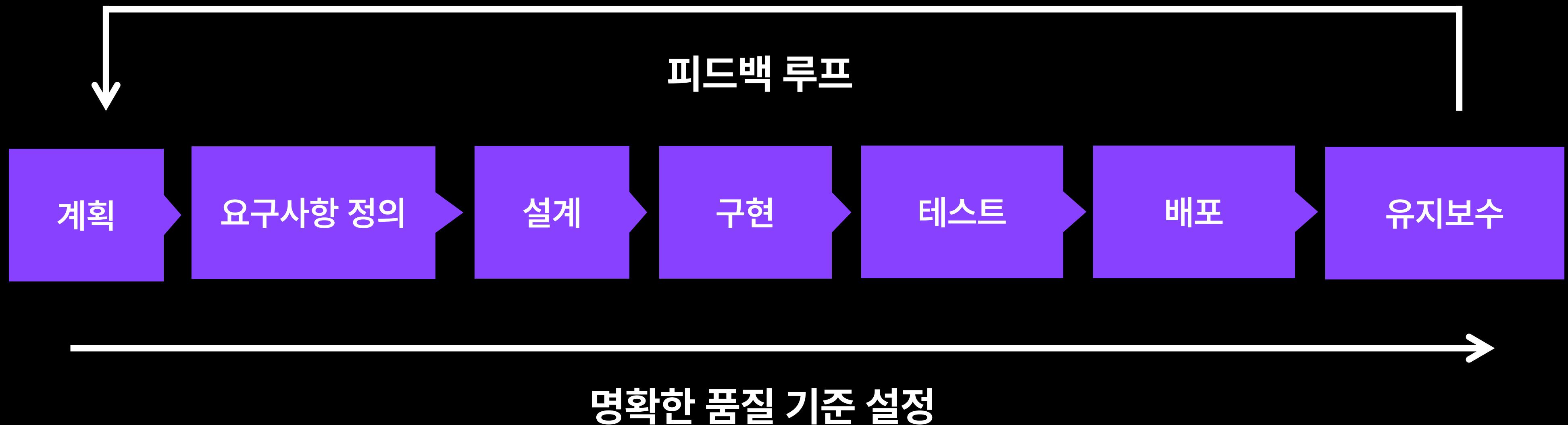
SDLC를 통한 효율적인 개발



+ SDLC



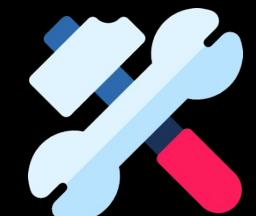
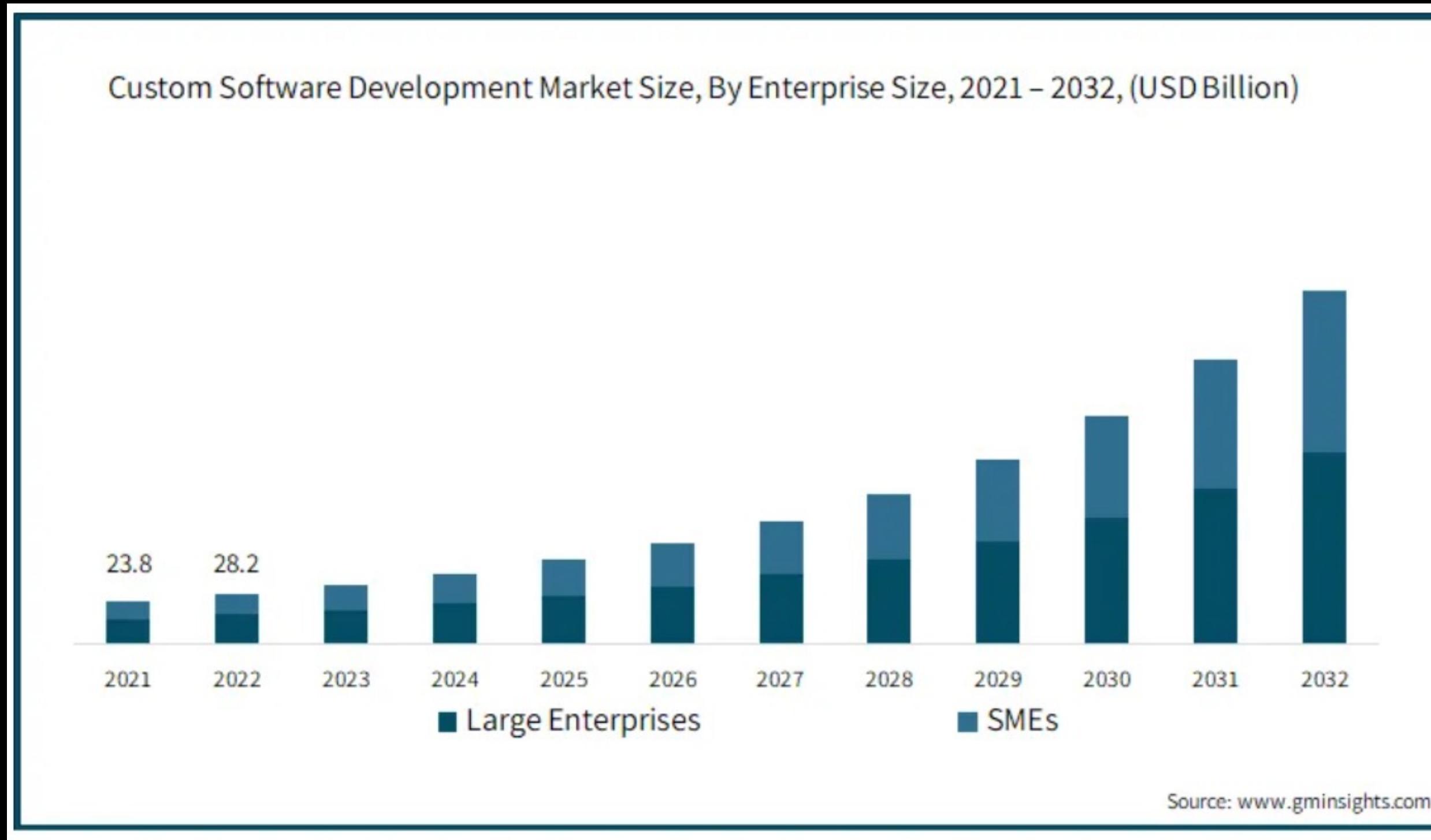
SDLC와 품질 보증





강력한 도구 SDLC

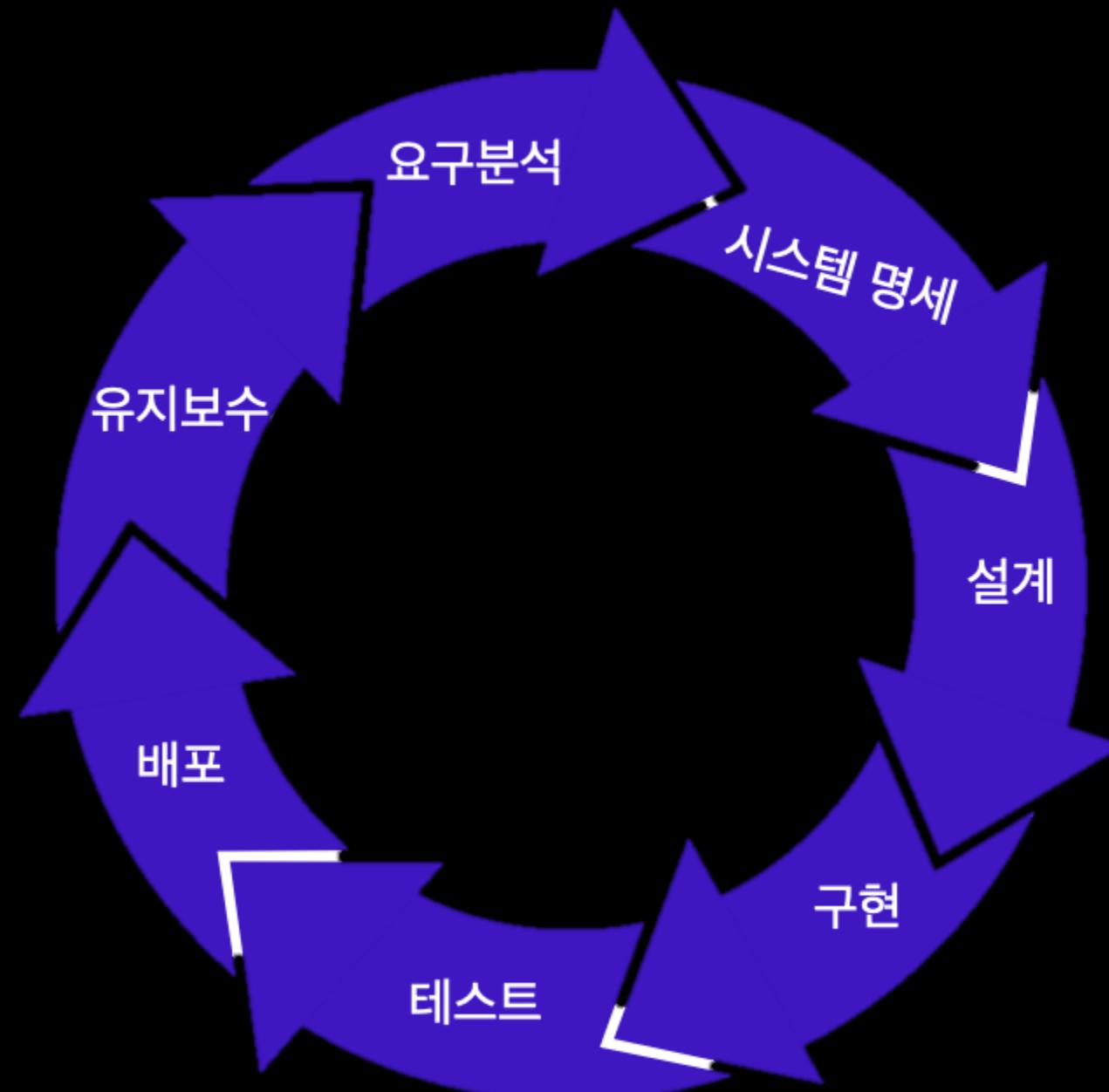
소프트웨어 개발 시장 동향



그를 뒷받침하는
강력한 도구, SDLC!



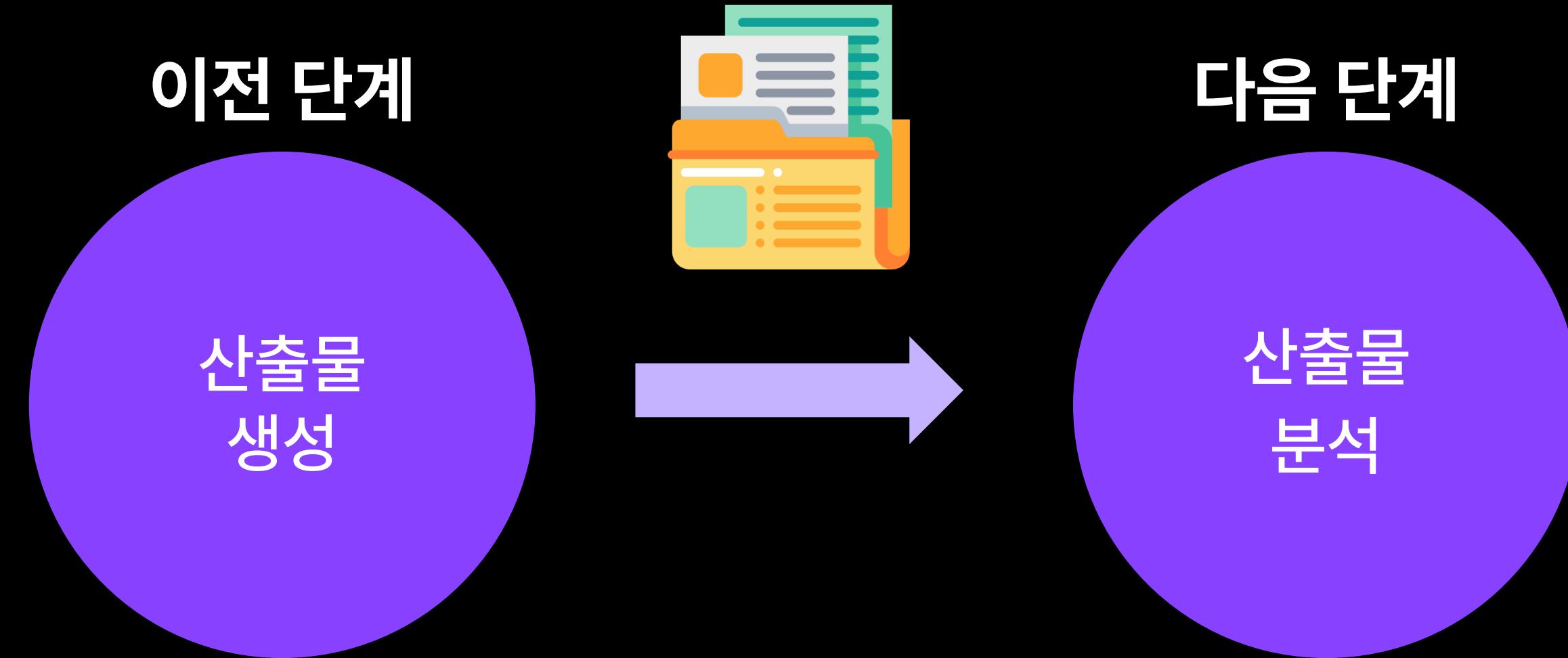
SDLC 단계 설명



1. 계획
2. 요구사항 정의
3. 설계
4. 구현
5. 테스트
6. 배포
7. 유지보수



단계 간 상호작용



이전 단계의 산출물을 다음 단계로 전달



계획 단계에서 하는 일



계획 단계 To-do

- ✓ 목표와 범위 정의
- ✓ 일정과 리소스 계획
- ✓ 위험 요소 분석



계획 단계 산출물

1. 프로젝트 개요

- 프로젝트 이름: [프로젝트 이름]
- 프로젝트 목적: [프로젝트의 최종 목표와 기대 효과를 간단히 기술]
- 프로젝트 기간: [시작일] ~ [종료일]
- 프로젝트 팀: [팀 이름/구성원]
- 관련 문서: [참고 문서 또는 링크]

2. 프로젝트 목표 및 범위

목표

- [핵심 목표를 구체적으로 나열]
 - 예: "사용자 친화적인 웹 플랫폼 개발"
 - 예: "주요 기능의 효율성을 20% 향상"

범위

- 포함된 작업: [프로젝트에 포함된 주요 업무]
 - 예: UI/UX 디자인, 백엔드 API 개발, 사용자 테스트.
- 제외된 작업: [프로젝트 범위를 벗어난 작업]
 - 예: 데이터 수집, 하드웨어 설계.

리스크 관리 보고서

1. 개요

- 프로젝트 이름: [프로젝트 이름]
- 작성자: [작성자 이름]
- 작성일: [작성일]
- 목적: 프로젝트의 성공적인 수행을 위하여 예상되는 위험 요소를 식별, 평가 및 관리 전략을 수립함.

2. 리스크 요약

리스크 ID	리스크 유형	리스크 설명	발생 가능성 (1~5)	영향도 (1~5)	우선순위
R-001	일정	개발 일정 지연 위험	4	5	높음
R-002	기술	신기술 도입 실패	3	4	중간
R-003	자원	인력 부족	4	4	높음

3. 주요 리스크 세부 정보

1. 리스크 ID: R-001

- 리스크 유형: 일정
- 설명: 특정 단계에서 요구사항 변경으로 인해 개발 일정이 지연될 가능성이 있음.
- 영향: 전체 프로젝트 완료 시점이 지연될 위험.

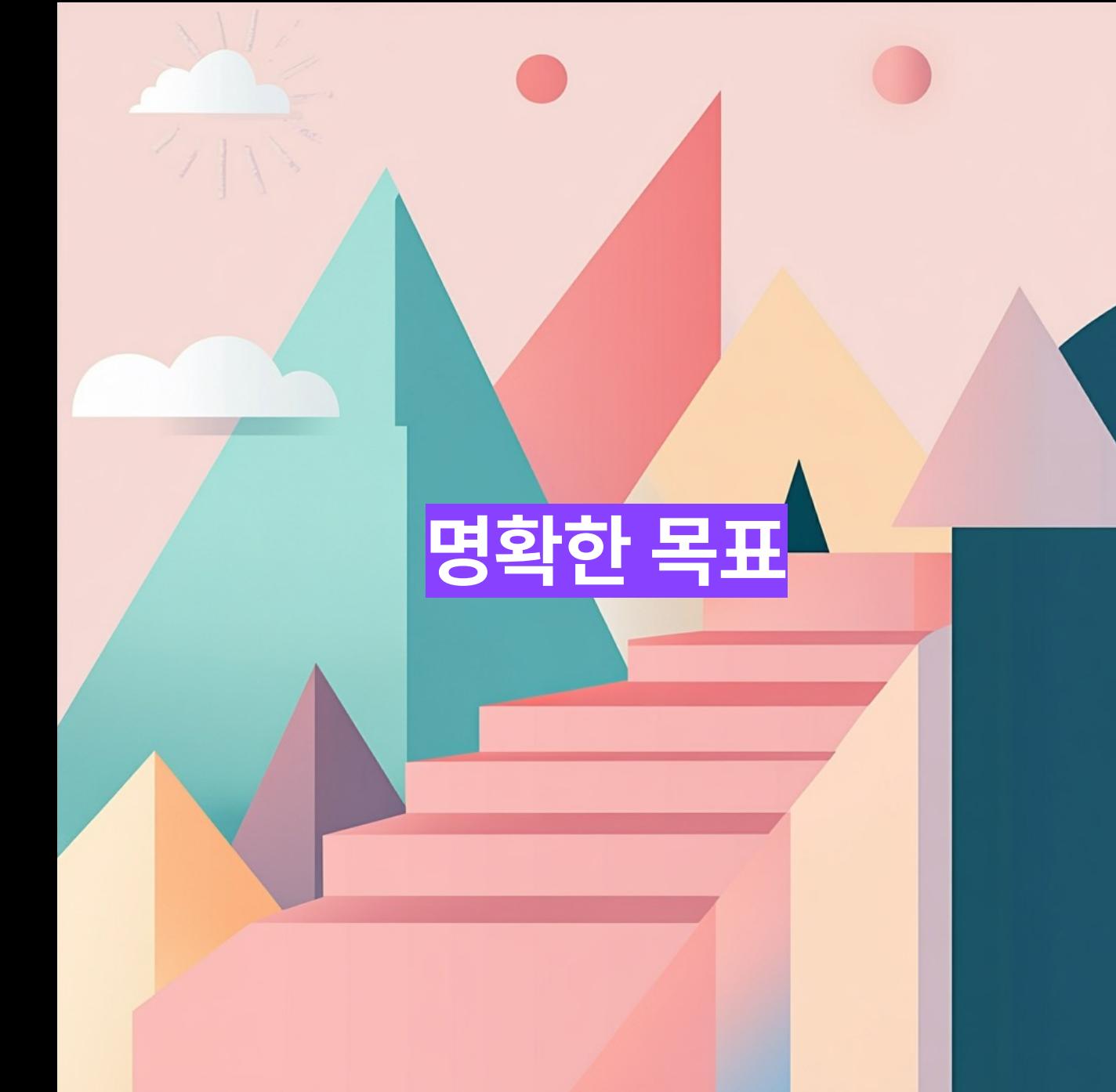
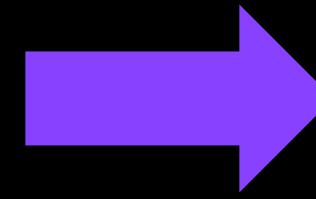
프로젝트 계획서 샘플

리스크 관리 보고서 샘플



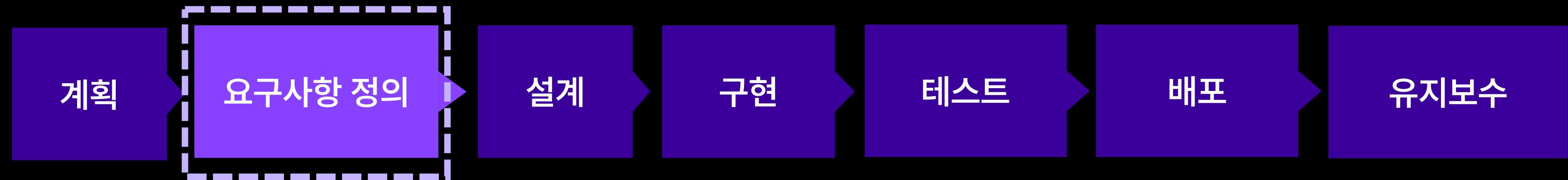
계획 단계의 중요성

탄탄한 계획





요구사항 정의 단계에서 하는 일



요구사항 정의 단계 To-do

- ✓ 요구사항 수집과 분석
- ✓ 요구사항 문서화
- ✓ 명확한 소통



요구사항 정의 단계 산출물

요구사항 ID	요구사항명	기능ID	기능명	상세 설명	필수 데이터	선택 데이터
MEM01	로그인	MEM01_LOGI_N01	자체 로그인 기능	우리 기능을 사용하기 위해 로그인 필수. (로그인 안했을 시 MAIN01, ADD01, ADD02, ADD03_ 눈팅)만 이용 가능)	ID, 비밀번호	
		MEM01_LOGI_N02	카카오 연동 로그인 기능	카카오 연동해서 아이디 비번 입력 없이도 쉽게 로그인 되도록 하는 기능		
		MEM01_LOGI_N03	아이디 찾기	전화번호와 이름을 입력하면 해당 아이디 보여주기	전화번호, 이름	-
		MEM01_LOGI_N04	비밀번호 재설정	입력 받은 전화번호로 임의의 새로운 비밀번호 전송, 후에 MEM03 기능을 통해 비밀번호 수정	전화번호, 아이디	-
MEM02	회원가입	MEM02_SIGN_UP01	자체 회원가입 기능	자체 회원가입 기능	이름, 주민번호 앞+뒤 1자리, 전화번호, 이메일, 현주소, ID, 비번	프로필 사진, 직업군인 여부, 임관일
		MEM02_SIGN_UP02	전화번호 인증	통신사, 전화번호, 악관 등의 체크 후 전화번호 인증 진행	통신사, 전화번호, 악관 등의	
		MEM02_SIGN_UP03	카카오 연동 회원가입 기능	카카오 연동 회원가입 기능, 추가 데이터 입력 필요	이름, 주민번호 앞+뒤 1자리, 전화번호, 이메일, 현주소, 악관 등의	직업군인 여부, 임관일
MEM03	회원 정보 수정	MEM03_MODI_FY01	회원 정보 수정	이름, 생년월일, 성별, 전화번호 변경 불가 다른 회원정보 수정 페이지		프로필사진, 이메일, 현주소, ID, 비밀번호, 직업군인 여부, 임관일
MEM04	회원 탈퇴	MEM04_DRO_P01	회원 탈퇴	필입 백스로 한번 더 확인, 비밀번호 입력 후 회원 탈퇴 (정보 전부 삭제)	비밀번호, 탈퇴 체크	
MEM05	로그아웃	MEM05_LOG_OUT01	로그아웃	로그아웃 기능		

유저 시나리오: 취미 정보를 제공받고 활동하는 사용자

장면				
시나리오	모즘 회사 생활도 그렇고 사는거 재미가 없다. 새로운 활동이라도 시작해서 살며 활력을 얻고 싶다. 예전에 재밌게 배웠던 도트임을 다시 시작 해보는 게 어떨까?	집과 가까운 곳 위주로 맘에 드는 취미 활동을 살펴 본다.	맘에 드는 활동을 몇 가지를 찾았는데 전부 다 해 볼 수는 없고 가격, 위치, 시간 등을 비교해서 가장 맘에 드는 것 하나를 골라야겠다.	맘에 드는 취미 활동을 골랐다. 그런데 나는 도트임 조로인데 다른 참가자들을 모두 실력자가 아닐지 고민 된다. 개설자는 나랑 잘 맞는 사람일까?
니즈	시간절약을 위해 되도록 집 가까운 곳에서 취미활동을 하고 싶다.	맘에 드는 수업들을 따로 모아보고 싶다.	맘에 드는 취미 활동 몇 가지는 간략하게 보여줘서 비교하고 싶다.	개설자와 다른 참가자들에 대한 정보를 통해 취미 활동을 결정하는데 도움을 얻고 싶다.
사용자 행동	취미 플레이너를 통해 집과 가까운 곳에서 할 수 있는 취미 활동을 찾기 위해 위치 서비스를 켜서 도트임 수업으로 상세 검색한다.	맘에 드는 몇 가지 수업에 좋아요를 누른다.	‘좋아요’ 페이지에 가서 취미 활동의 위치와 가격등 간략한 정보를 비교한다.	개설자의 페이지에서 개설자의 삶에 정체와 흥미, 그리고 다른 참가자들의 정보를 보고 결정한다.
기능	<ul style="list-style-type: none"> 위치 기반 검색 후 상세 검색 기능 	<ul style="list-style-type: none"> 취미 수업에 좋아요를 누를 수 있는 기능 	<ul style="list-style-type: none"> 좋아요 한 취미 활동의 간략한 정보를 제공하는 ‘찜한 내역’ 페이지 제공 	<ul style="list-style-type: none"> 개설자 페이지에 흥미 정보 제공, 수업 상세페이지에 참여자 정보를 수 있는 특별 제공

요구사항 명세서 샘플

사용자 시나리오 샘플

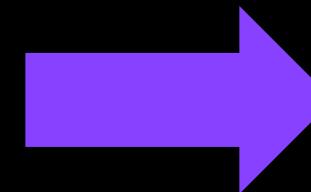
<https://velog.io/@juyeon/%EC%9A%94%EA%B5%AC%EC%82%AC%ED%95%AD-%EC%A0%95%EC%9D%98%EC%84%9C-%EC%9E%91%EC%84%B1%ED%95%98%EB%8A%94-%EB%B2%95>

<https://medium.com/@hyey1993/ux-practice-14-%EC%9C%A0%EC%A0%80-%EC%8B%9C%EB%82%98%EB%A6%AC%EC%98%A4-76a64d6147fc>



요구사항 정의 단계의 중요성

“좋아하는 맛집을 즐겨찾기 하고 싶어요 ”



찜

배달·포장

장보기·쇼핑

총 10개

파스타예요 강동천호본점

강동 최초 반반 메뉴 파스타&필라프, [[첫...]

최소주문 3,000원

30% 쿠폰

쿠폰

그램치킨 강동직영점

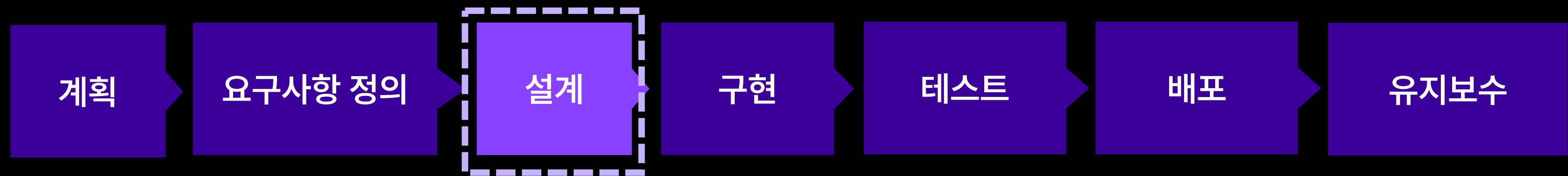
★ 4.9 (100+) [[주문수 1위]] 양념 치킨라이스, [...]

최소주문 5,900원

방향성 제시



설계 단계에서 하는 일



설계 단계 To-do

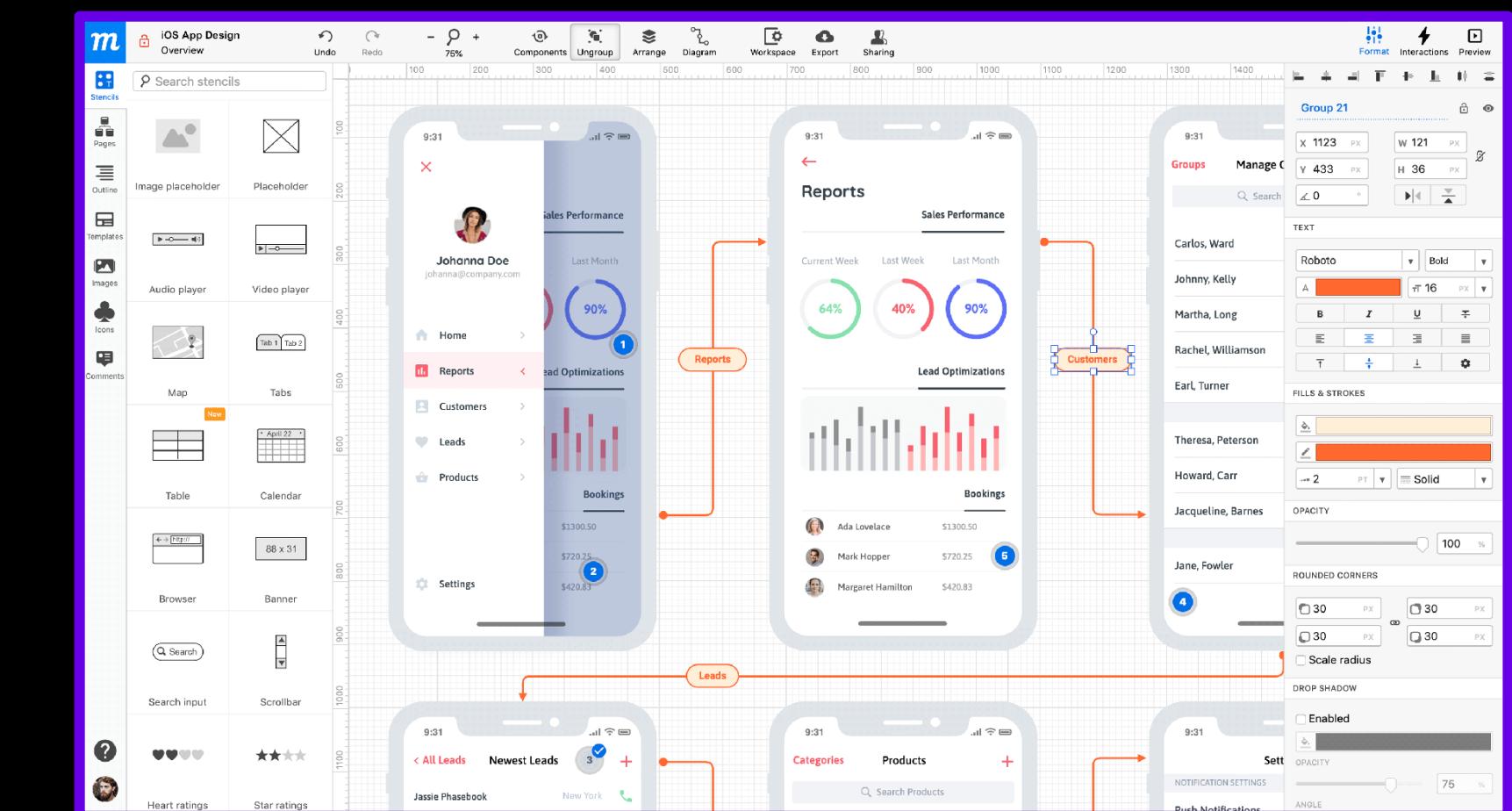
- ✓ 아키텍처 설계
- ✓ 데이터 흐름 정의
- ✓ 기술적 문제 해결



설계 단계 산출물

테이블정의서		Database											
		Schema		생성일		2013년 02월							
테이블명													
COMMENT													
Col #	Column Name	Data Type	Key	Null?	Identity	Default	Comments						
1	seq	int		N			시퀀스						
2	chance_id	varchar(50)		Y			ID						
3	tid	varchar(80)		Y			거래번호(자동 생성)						
4	ResultCode	varchar(4)		Y			결과코드(-_-)						
5	ResultMsg	varchar(300)		Y			결과 내용 메시지						
6	Moid	varchar(80)		Y			상점 사용 주문번호						
7	ApplDate	varchar(8)		Y			승인날짜						
8	ApplTime	varchar(6)		Y			승인시각						
9	ApplNum	varchar(25)		Y			승인번호						
10	PayMethod	varchar(20)		Y			지불방법(- - - - -)						
11	TotPrice	varchar(25)		Y			결제결과금액						
12	card_num	varchar(30)		Y			카드번호						
13	card_interest	varchar(4)		Y			할부여부(1이면 무이자할부)						
14	card_quota	varchar(4)		Y			할부기간						

데이터베이스 설계서 샘플



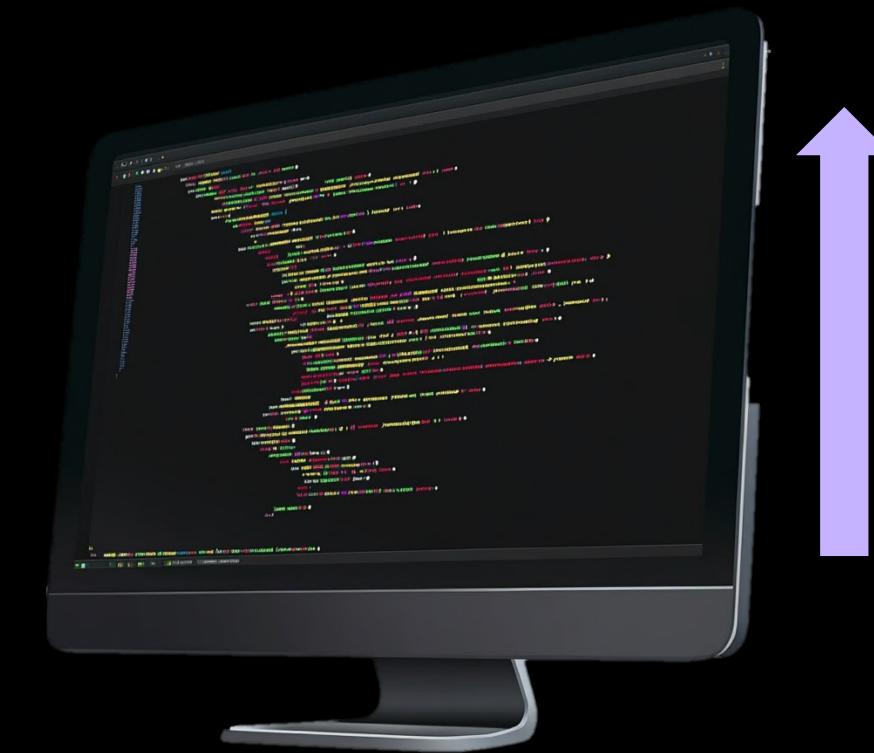
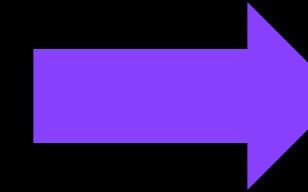
UI 프로토타입 샘플

https://yamestyle.com/281#google_vignette
<https://u-and-u.tistory.com/52>



설계 단계의 중요성

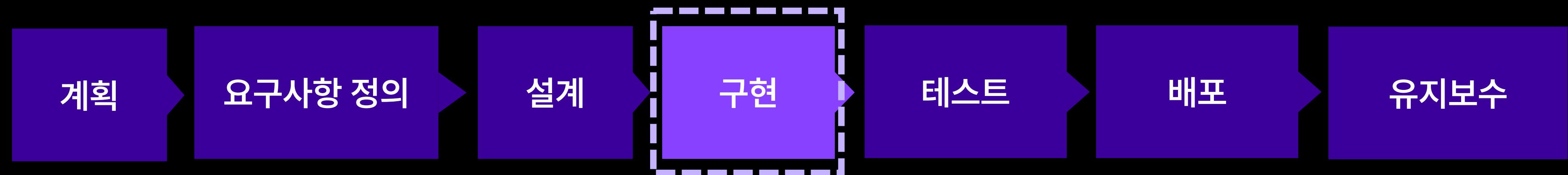
세부적인 설계



개발 속도 향상



구현 단계에서 하는 일

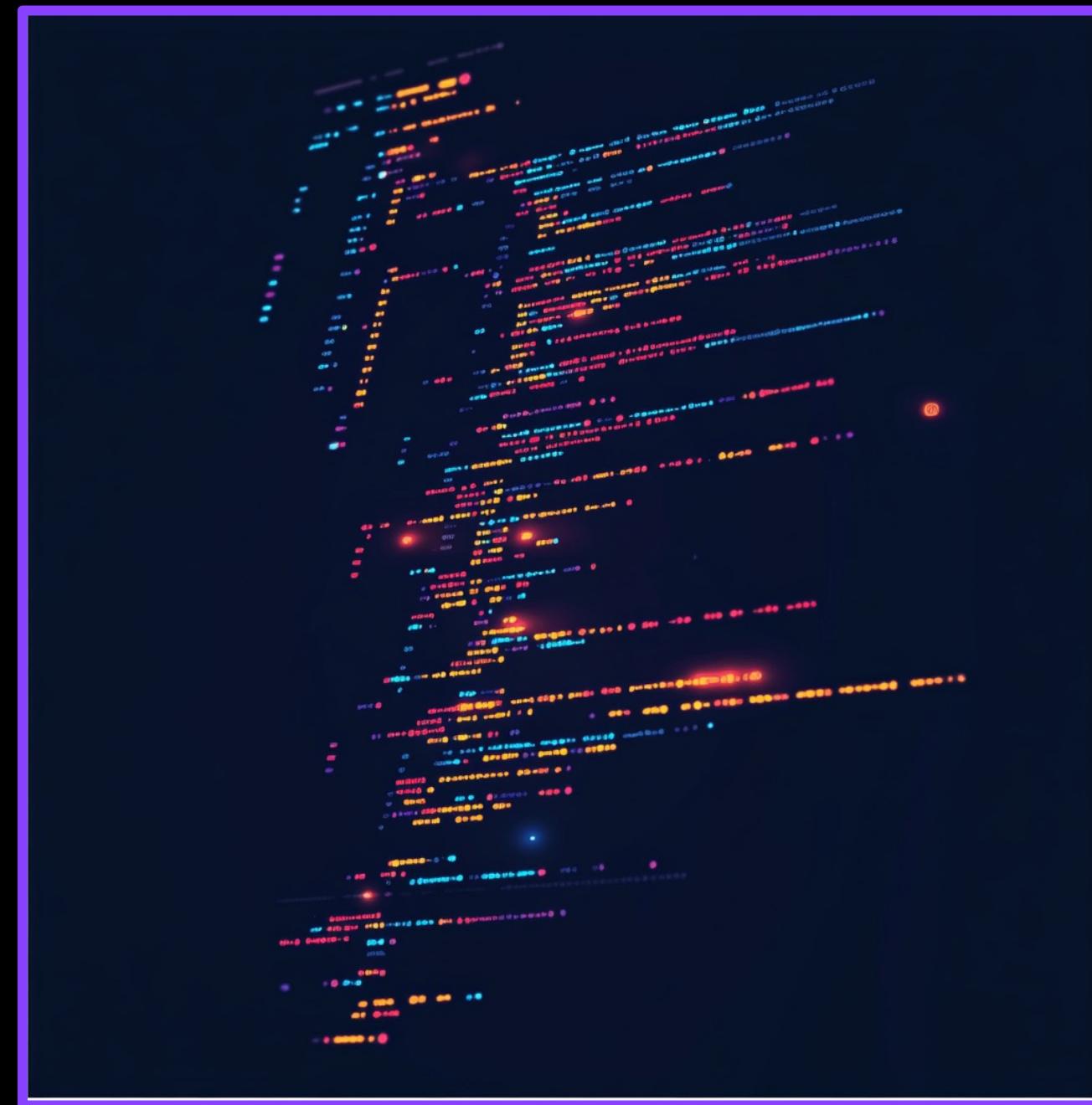


구현 단계 To-do

- ✓ 설계 기반 개발
- ✓ 코딩 작업
- ✓ 빌드 작업



구현 단계 산출물



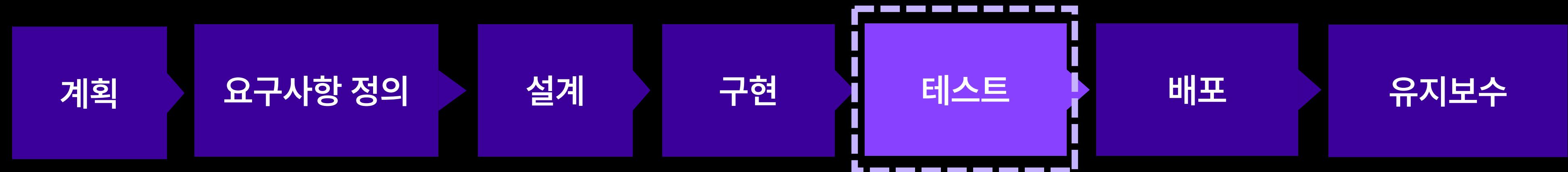
소스 코드



실행 가능한 애플리케이션



테스트 단계에서 하는 일



테스트 단계 To-do

- ✓ 기능 및 성능 검증
- ✓ 결함 발견 및 수정
- ✓ 주요 활동



테스트 단계 산출물

1. 테스트 대상 소개

가. ConcourseSuite(v6.0)

웹 기반의 CRM 솔루션으로 전신은 Centric-CRM이며, 공식적으로 데이터베이스는 PostgreSQL 8.x를 사용하고, 애플리케이션 서버로는 Apache Tomcat 6.0을 사용함

* 출처 : 공개SW TRM-솔루션 프로파일 “솔루션 설명” 참조

2. 테스트 케이스 및 시나리오

ConcourseSuite의 신뢰성을 검증하기 위하여 테스트 케이스에 기반을 둔 기능 테스트와 테스트 시나리오에 기반을 둔 비 기능 테스트를 수행한다.

가. 기능별 테스트케이스 현황

[표 2-1. 기능별 테스트케이스 현황]

기 능	테스트 케이스 수
설치 및 삭제	3
실행	3
로그인	6
연락일정관리	4
모임일정관리	-

테스트 결과 보고서 샘플

〈표 1-12〉 단계별 결함 유형 정의 예시

단계별 결함 유형	주요 내용
분석	요구사항의 표준 미준수, 테스트 불가능, 불명확, 불완전, 불일치, 기타 결함
설계	설계 표준 미준수, 테스트 불가능, 불명확, 불완전, 불일치, 기타 결함
개발	개발 표준 미준수, 불완전, 불일치, 인터페이스 결함, 데이터 결함, 기타 결함
테스트	업무 이해 부족, 팀 간 의사소통 부족, 개발자 실수

(2) 〈표 1-13〉과 같이 발생한 결함에 대한 결함 상태를 정의한다.

〈표 1-13〉 결함 상태 정의

결함 상태	설 명
Open	결함이 보고되었지만 아직 분석되지 않은 상태
Assigned	영향 분석 및 수정을 위해 결함을 개발자에게 할당한 상태
Fixed	개발자가 결함을 수정한 상태
Closed	결함이 수정되었는지 확인하고 회귀 테스트 시 결함이 발견되지 않은 상태. 수정된 결과가 만족스럽지 않을 경우, 결함의 상태를 'open'으로 변경한다.
Deferred	결함 우선순위가 낮게 분류되었기 때문에 결함 수정을 연기한 상태
Clarified	보고된 결함이 프로젝트 팀에 의해 결함이 아니라고 판단된 상태

(3) 정의한 결함 유형을 이용하여 결함관리 현황을 [그림 1-9]와 같이 결함 상태를 작성한다.

결함 목록

<https://velog.io/@tisalstjr58/%EC%95%A0%ED%94%8C%EB%A6%AC%EC%BC%80%EC%9D%B4%EC%85%98-%ED%85%8C%EC%8A%A4%ED%8A%B8-%EC%88%98%ED%96%89-3741p>



퀴즈 타임



QUIZ 1번부터 7번을 풀어봐요!

계획 단계부터 시작하는 SDLC 여정



계획 단계는 구체적으로 어떤 업무를 하는 걸까요?



프로젝트 목표 정의

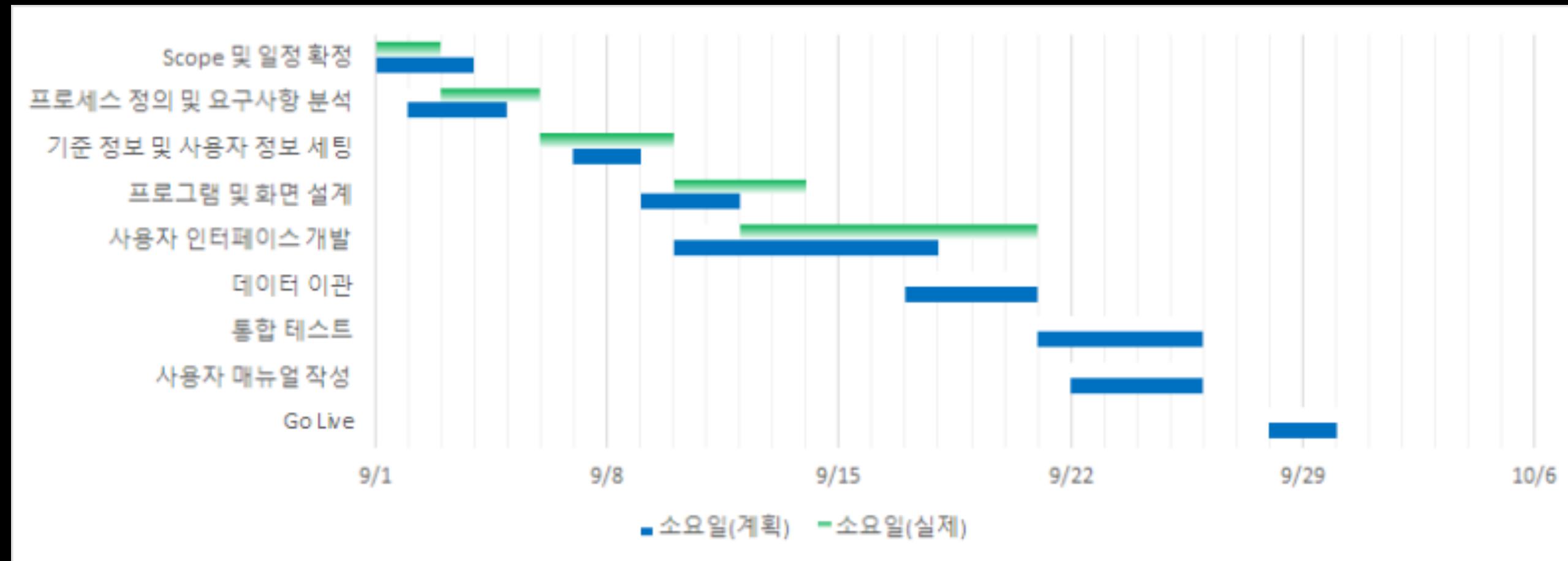
SMART 목표 설정 기준





리소스와 일정 계획

[간트(Gantt) 차트]



- 필요 인력, 예산, 장비 및 배분
- 팀 구성과 역할 분담
- 주요 *마일스톤 설정
- 작업 우선순위와 일정 조율

*마일스톤 : 프로젝트 진행 중 중요한 목표나 주요 단계의 달성을 나타내는 이정표



위험 평가 및 관리

[리스크 매트릭스]

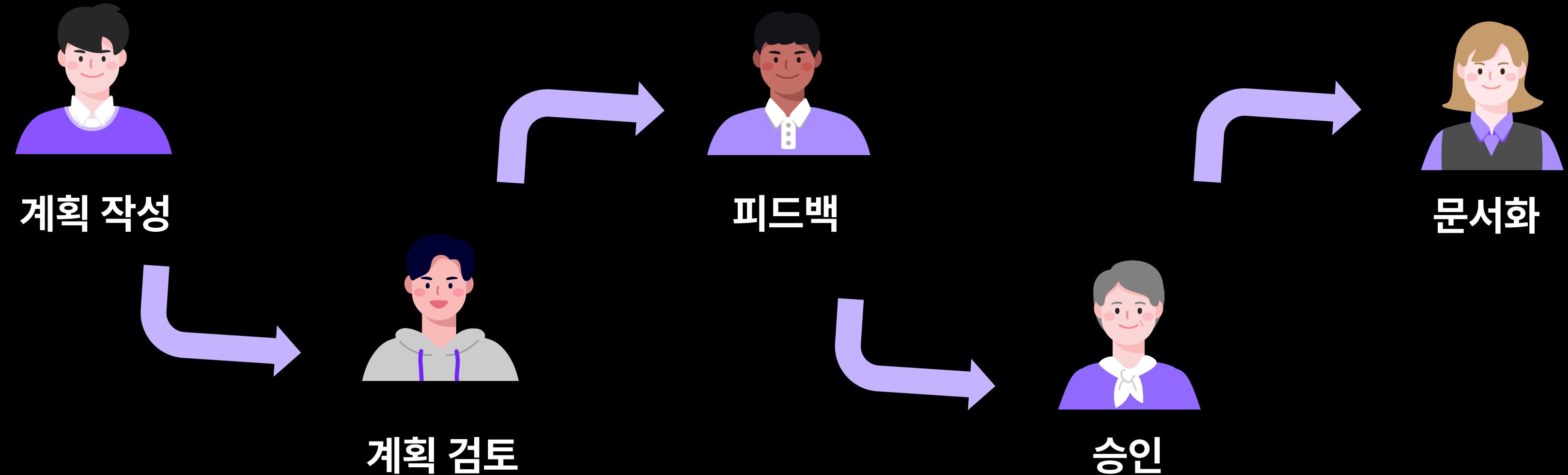
Risk Level(위험 수준) = Likelihood(발생 빈도, 가능성) * Consequence/Impact(발생의 결과)

		Impact				
		Negligible	Minor	Moderate	Significant	Severe
Likelihood	Very Likely	Low Med	Medium	Med Hi	High	High
	Likely	Low	Low Med	Medium	Med Hi	High
	Possible	Low	Low Med	Medium	Med Hi	Med Hi
	Unlikely	Low	Low Med	Low Med	Medium	Med Hi
	Very Unlikely	Low	Low	Low Med	Medium	Medium

1. 리스크 식별
2. 리스크 평가 및 우선순위 지정
3. 리스크 대응 계획 수립
4. 리스크 모니터링 및 검토
5. 리스크 대응



계획 승인 절차





Key Performance Indicator (핵심 성과 지표)

주요 성과 지표(KPI)

- 계획 완료율: 설정된 목표 대비 달성을률
- 리스크 식별률: 식별된 위험의 수와 효과적인 대응 여부
- 자원 할당 정확도: 인력과 예산의 적절성 평가



퀴즈 타임



QUIZ 8번부터 9번을 풀어봐요!



요구사항 정의, 왜 중요한가요?



계획 단계에서 수립한 목표와 범위를 **구체화하는** 과정



사용자 요구사항 수집 방법

수집 방법	설명	장점	단점	활용 예시
인터뷰	이해관계자와 직접 대화 하여 요구사항을 심층적으로 탐구	구체적이고 심층적인 정보 수집 가능	시간이 오래 걸리고 주관적인 의견이 포함될 수 있음	주요 의사결정자와의 요구사항 탐색
설문조사	여러 사용자를 대상으로 표준화된 질문 을 통해 정보를 수집	많은 사람의 의견을 빠르게 수집 가능	응답의 깊이가 부족할 수 있음	다양한 사용자 그룹의 의견 수집
워크숍	이해관계자와 팀원들이 함께 참여 하여 협력적으로 요구사항 도출	협업을 통해 창의적인 아이디어와 합의 도출 가능	준비와 실행에 시간이 많이 소요됨	새로운 시스템의 기능 도출
관찰	사용자의 실제 작업 환경을 분석 하여 요구사항 파악	사용자가 요구사항을 명확히 설명하지 못할 때 유용	많은 시간과 노력이 필요함	고객 서비스 작업 흐름 분석
프로토타입	기본 모델을 개발하여 사용자 피드백 을 통해 요구사항 수정 및 보완	실제 사용 사례를 기반으로 요구사항을 구체화 가능	초기 개발 비용이 증가할 수 있음	사용자 인터페이스 요구사항 개발



요구사항 분석 기법



분석 도구 : UML, 요구사항 관리 도구



UML이란?

UML

Unified Modeling Language

- 개발을 위한 설계도
- 개발에 필요한 각 구성요소 간의 관계를 도식화



UML

Unified Modeling Language

장점	단점
<ul style="list-style-type: none">• 표준화된 소통 수단• 시각화• 재사용성 및 유지보수 용이• 문서화• 설계 오류 감소	<ul style="list-style-type: none">• 학습 곡선• 과도한 세부 사항• 시간 및 비용• 유연성 부족• 표현의 한계



UML

Unified Modeling Language

구조 다이어그램
(Structure Diagram)

행위 다이어그램
(Behavior Diagram)



UML

Unified Modeling Language

구조 다이어그램

(Structure Diagram)

클래스 다이어그램

컴포넌트 다이어그램

객체 다이어그램

프로필 다이어그램

복합체 구조 다이어그램

배치 다이어그램

패키지 다이어그램

행위 다이어그램

(Behavior Diagram)

활동 다이어그램

유즈케이스 다이어그램

상태 머신 다이어그램

시퀀스 다이어그램

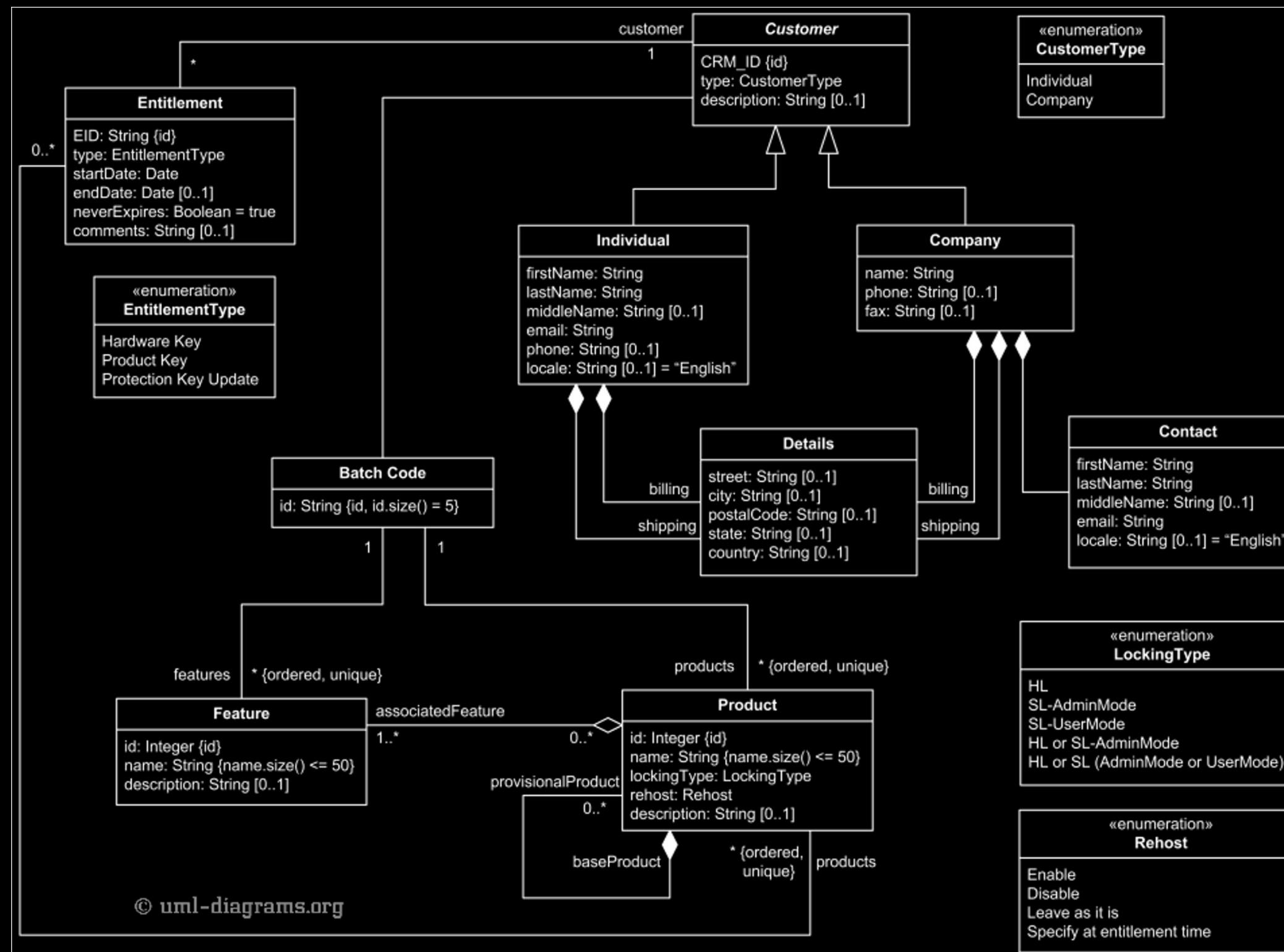
통신 다이어그램

상호작용 개요 다이어그램

타이밍 다이어그램



클래스 다이어그램



각 클래스의 필드와 메소드, 클래스 간의

관계를 도식화한 다이어그램

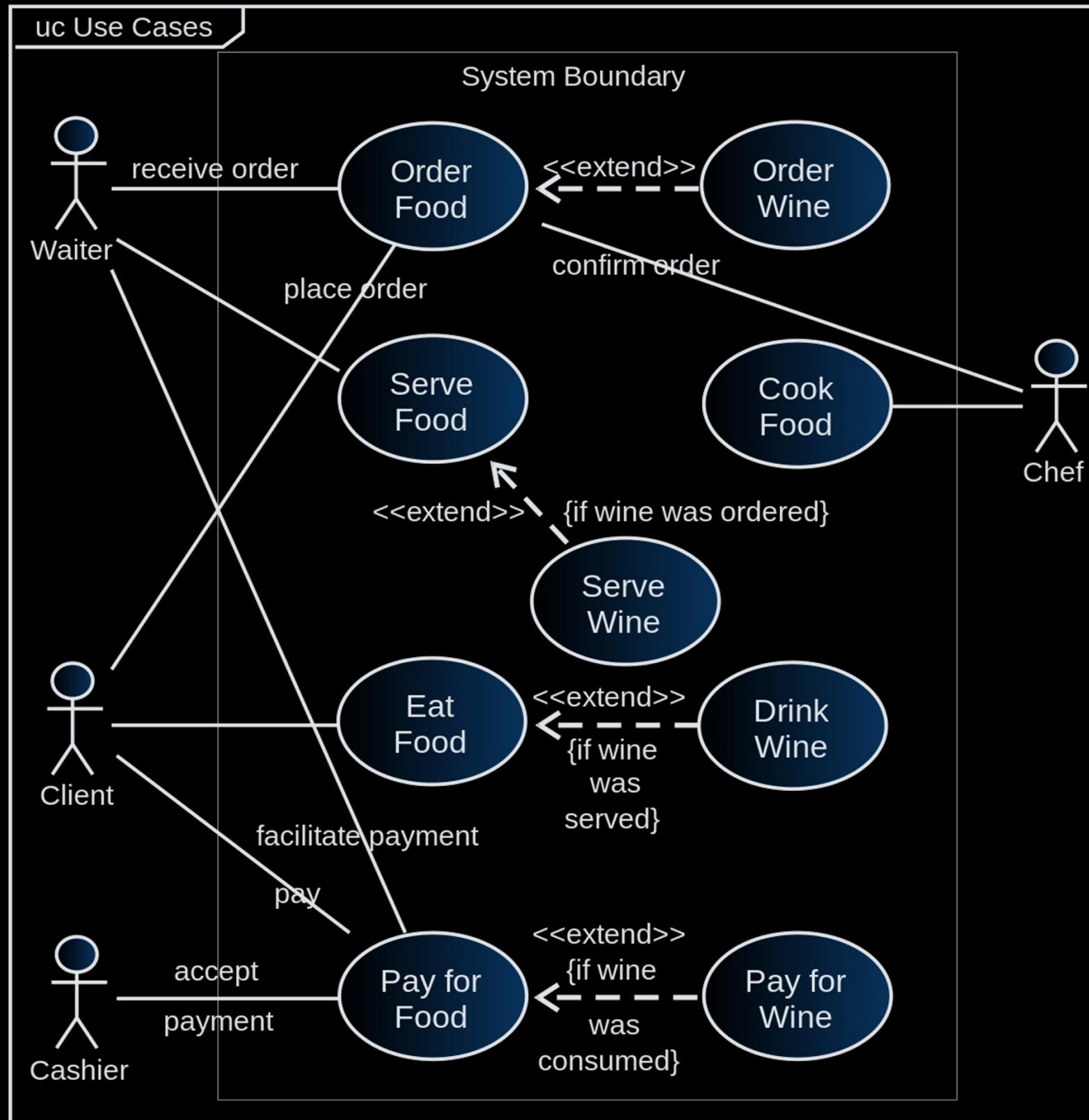
- 클래스 : 설계도

- 필드 : 설계도에 적힌 특징 / 속성

- 메소드 :설계도에 적힌 기능



유즈케이스 다이어그램



시스템이 제공해야 할 기능과
이를 사용하는 사용자(액터)들 사이의
상호작용을 시각적으로 표현하는 다이어그램



요구사항 문서화 기준

요구사항정의서

0000 프로젝트 구축
문서업데이트 : 2021.07.30

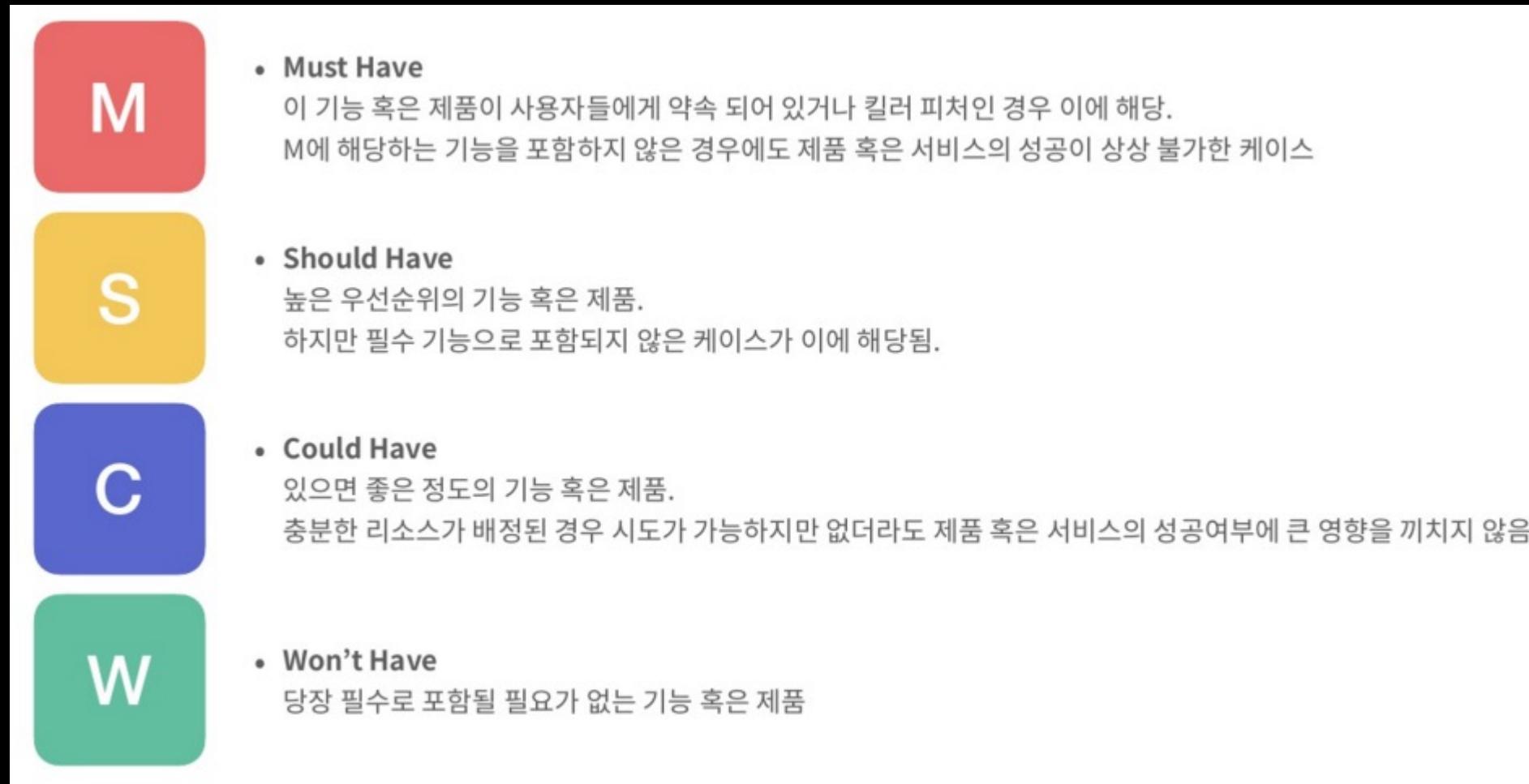
No	구분	요구사항 ID	요구사항 명	요구사항 상세 설명	요청자 (요청부서)	비고
1	웹	REQ-001	메인 앱 다운로드	- 메인 상단에 앱 다운로드 버튼 - 메인 하단에 앱 설치 주소 문자로 받을 수 있도록 구현 요청 (안드로이드/ iOS 구분 없이 휴대폰 번호로 다운로드 URL 링크를 전송하고 사용자가 링크 접속 시 사용자 OS에 따라 각 스토어 화면으로 이동할 수 있도록 구현)	홍길동 차장	https://goodrichapp.com/main 굿리치 웹사이트 참고
2	웹	REQ-002	로그인 (일반 회원)	일반 회원의 경우 아이디와 비밀번호로 로그인 할 수 있도록 함 <i>*주후 아이디/비밀번호 생성 정책 공유 예정</i>	신창구 과장	
3	웹	REQ-003	로그인 (소셜 회원)	소셜 회원의 경우 [네이버], [카카오], [구글] 3가지 소셜 수단으로 로그인 할 수 있도록 함	신창구 과장	
4	웹	REQ-004	아이디 찾기 (일반 회원)	일반 회원의 경우 아이디 찾기 기능을 제공 → 회원 본인 이름, 휴대폰 번호, 이메일 주소를 입력하고 일치할 경우 아이디 정보를 제공함	신창구 과장	
5	웹	REQ-005	비밀번호 재설정 (일반 회원)	일반 회원의 경우 비밀번호 재설정 기능을 제공 → 회원 본인 아이디, 이메일 주소를 입력하고 일치할 경우 비밀번호 재설정 인증 메일 발송 → 인증 메일을 통해 비밀번호를 재설정하도록 함	신창구 과장	메일 발송 대행 업체 확인 비밀번호 재설정 인증 메일 유효시간 확인 필요 (ex. 1시간 유효)
6	웹	REQ-006	GNB 메뉴 - 서비스 안내	중개 플랫폼 서비스에 대한 설명 페이지로 구성 (진근한 일러스트로 디자인)	홍길동 차장	
7	웹	REQ-007	GNB 메뉴 - 이동방법	실제 앱 화면으로 앱을 사용하는 튜토리얼을 제공하는 페이지로 구성	홍길동 차장	
8	웹	REQ-008	GNB 메뉴 - 고객센터	- 제공하는 공지사항을 확인할 수 있는 메뉴 - 관리자가 공지사항을 등록/수정할 수 있는 기능 - 제공하는 FAQ(자주 묻는 질문)를 확인할 수 있는 메뉴 - 관리자가 FAQ를 등록/수정할 수 있는 기능 - 1:1 문의를 통해 질문할 수 있는 메뉴 - 관리자가 답변을 등록/수정할 수 있는 기능	홍길동 차장	보드자료/이벤트 등의 카테고리 추가 검토 필요
9	앱	REQ-009	앱 스크래시	앱의 데이터를 로딩하는 동안 보여줄 스크래시 스크린 필요	마케팅팀	관리자 기능이 필요할지 확인 필요 - 스크래시 이미지 교체 - 스크래시 노출 시간
10	앱	REQ-010	게이트 페이지	보호자/간병인 중 한 가지를 선택할 수 있는 게이트 페이지 필요	홍길동 차장	
11	앱	REQ-011	로그인 (일반 회원)	일반 회원의 경우 아이디와 비밀번호로 로그인 할 수 있도록 함 <i>*주후 아이디/비밀번호 생성 정책 공유 예정</i>	신창구 과장	
12	앱	REQ-012	로그인 (소셜 회원)	소셜 회원의 경우 [네이버], [카카오], [구글], [애플] 4가지 소셜 수단으로 로그인 할 수 있도록 함 <i>*iOS의 경우 애플 로그인 추가</i>	신창구 과장	
13	앱	REQ-013	메인 - 둘러보기	회원가입/로그인 없이 앱을 둘러볼 수 있는 둘러보기 기능 필요	마케팅팀, 신창구 과장	회원가입/로그인 없이 둘러볼 수 있는 메뉴들 정리 필요 → 로그인이 필요한 메뉴 접근 시 열렸 OR 로그인 페이지로 이동 처리?
14	앱	REQ-014	메인 - 배너	관리자가 등록한 배너가 앱 메인에 노출되어야 함 (보호자/간병인 서로 다른 배너를 등록할 수도 있음)	마케팅팀	노출될 배너 위치 정의 필요

- 간결성
- 추적 가능성
- 이해 가능성

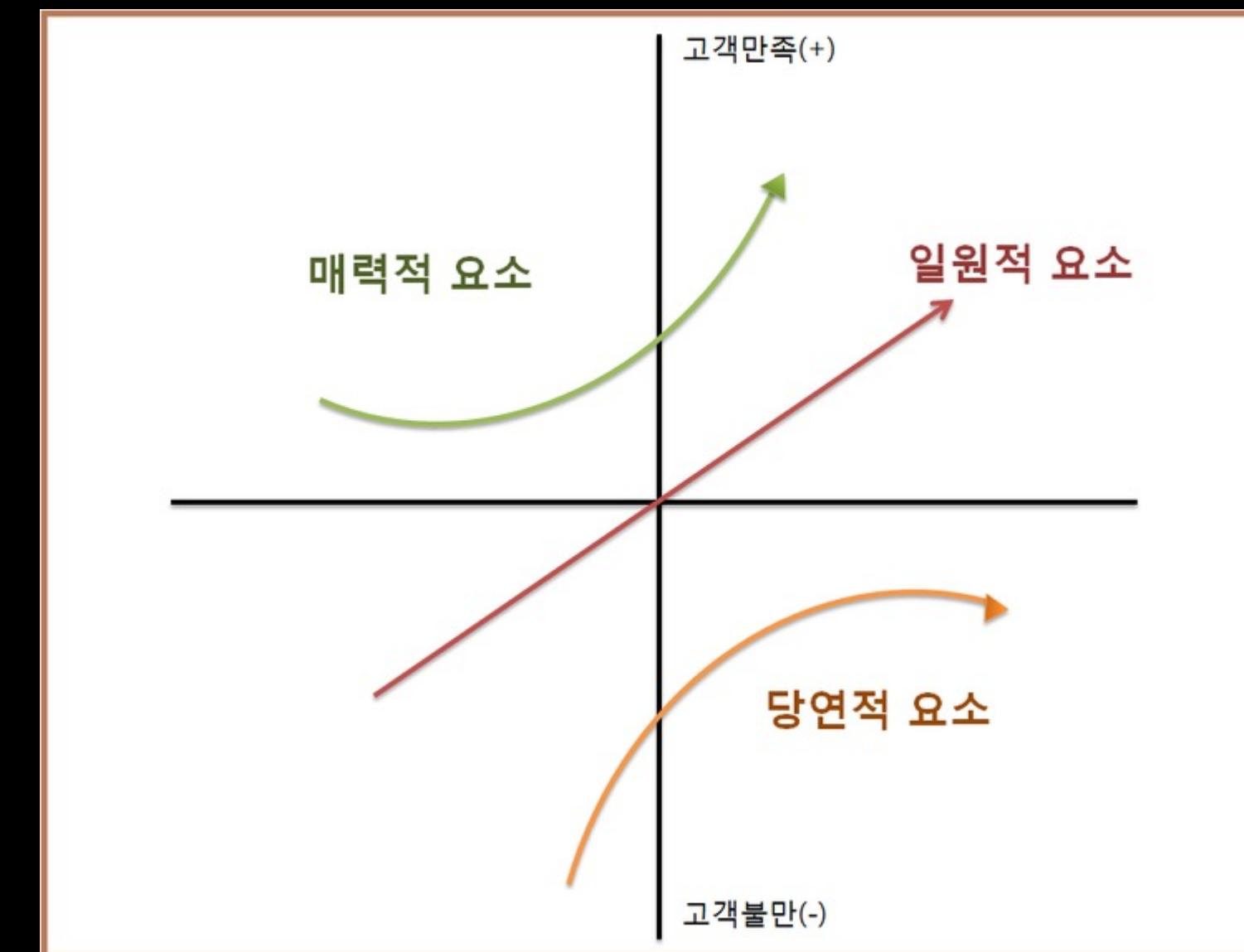


요구사항 우선순위화 기법

MoSCow 기법



Kano 모델

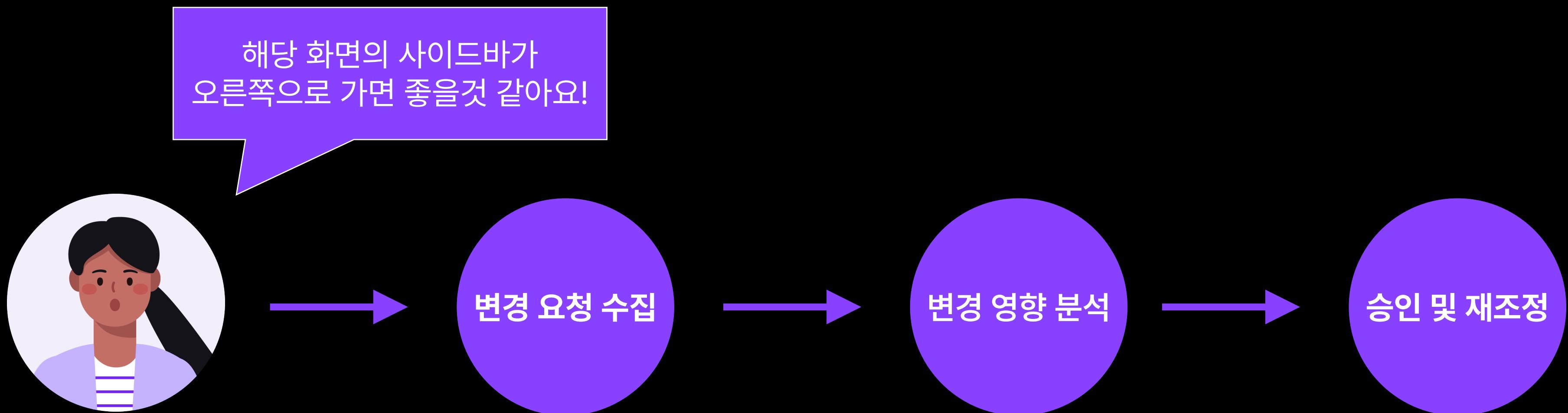


<https://blog.naver.com/germancert1/220489674734>

<https://blog.naver.com/90kimjieun/223102729822>



요구사항 변경 관리





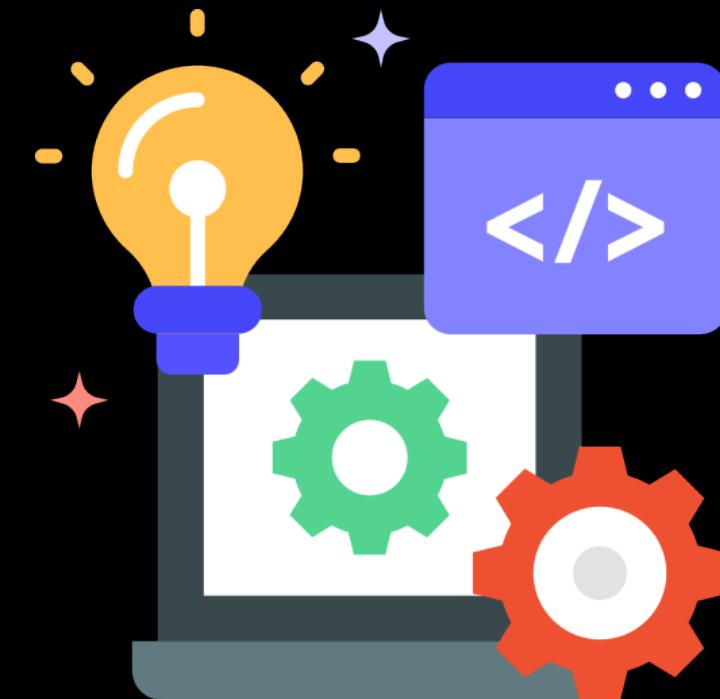
퀴즈 타임



QUIZ 10번부터 11번을 풀어봐요!



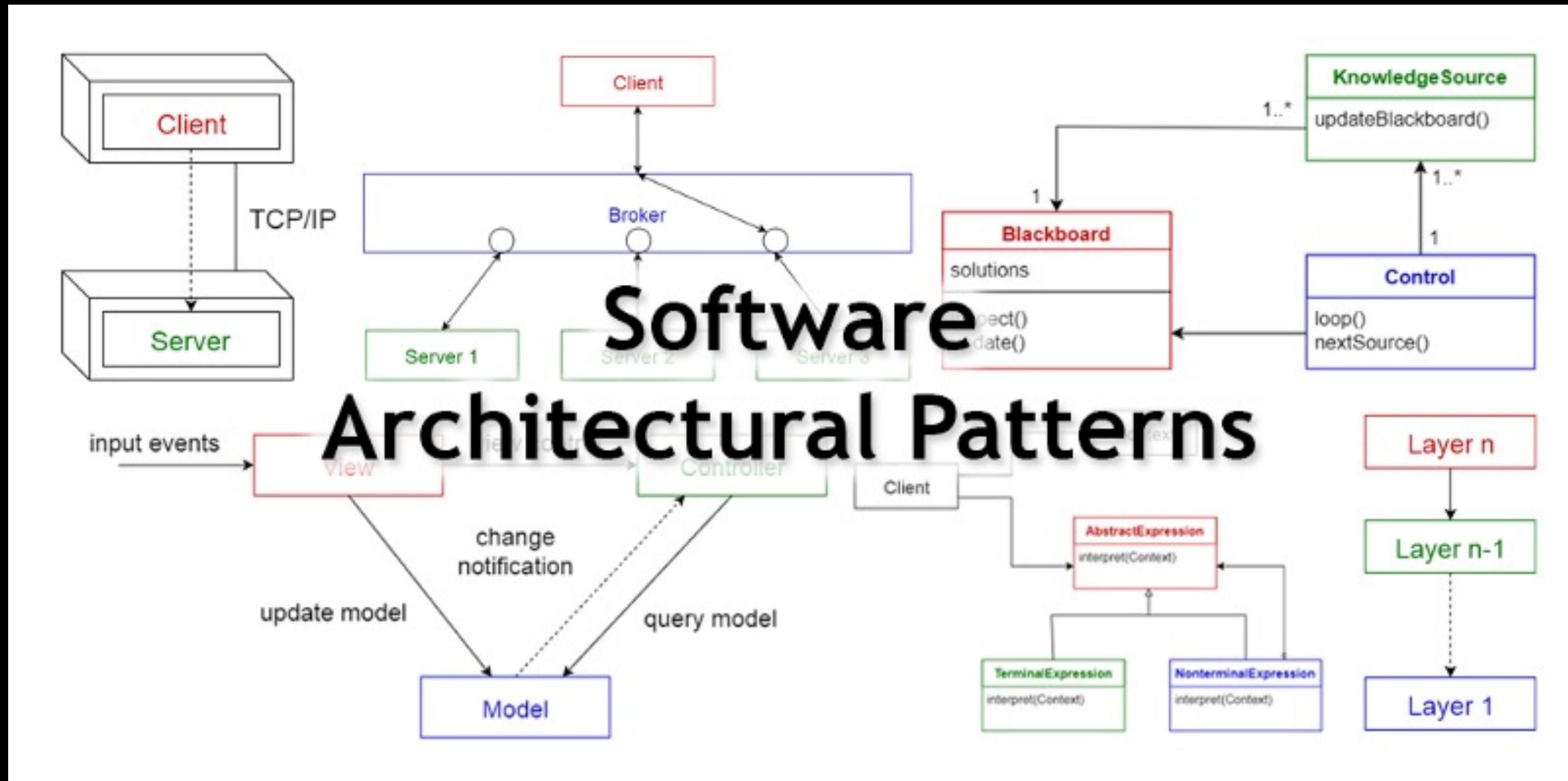
설계 단계, 소프트웨어의 청사진



요구사항 정의 단계에서 **도출된 정보**를 바탕으로
소프트웨어의 **구조와 동작 방식**을 구체화



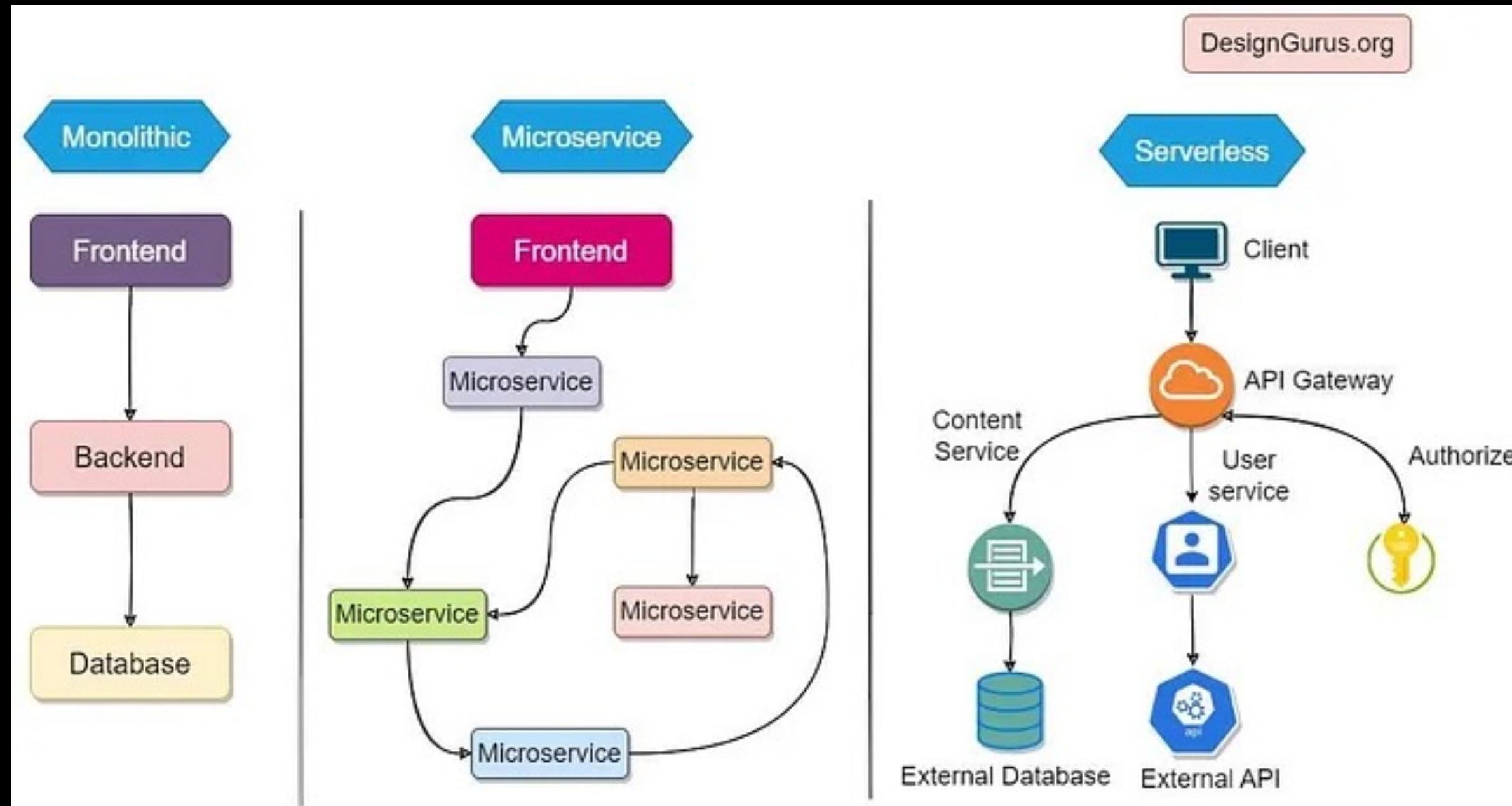
소프트웨어 아키텍처란



소프트웨어 시스템의 전반적인 구조와 설계를 정의하는 틀, 지침서



소프트웨어 아키텍처 설계 유형

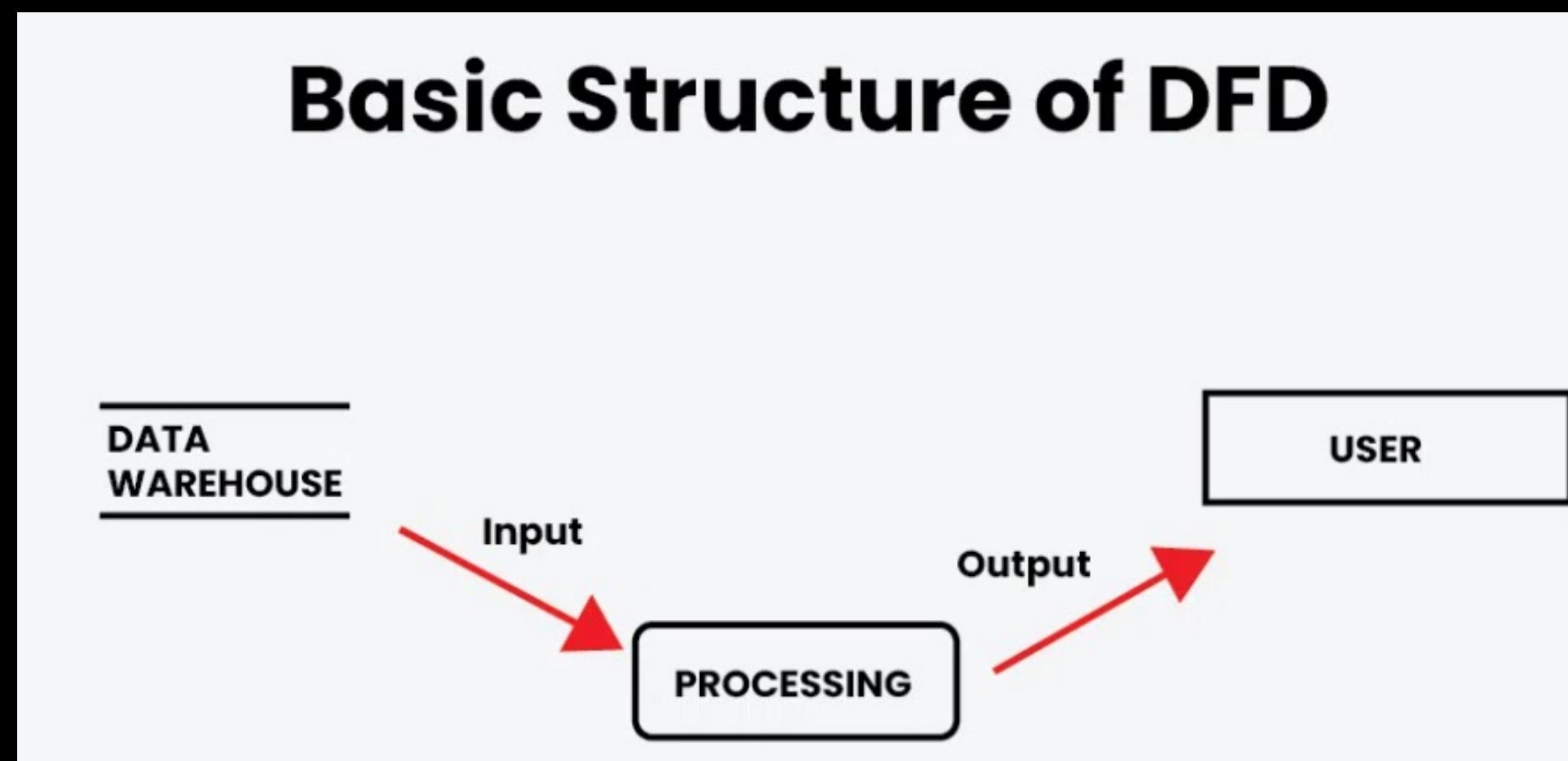


1. **계층형 아키텍처:** 프론트엔드, 백엔드, 데이터베이스를 계층적으로 분리
2. **마이크로서비스 아키텍처:** 독립적 서비스로 구성되어 유연성과 확장성이 높음
3. **서버리스(Serverless):** 클라우드 플랫폼을 활용하여 서버 관리 부담 감소



데이터 흐름 설계

[데이터 흐름 다이어그램(DFD)]

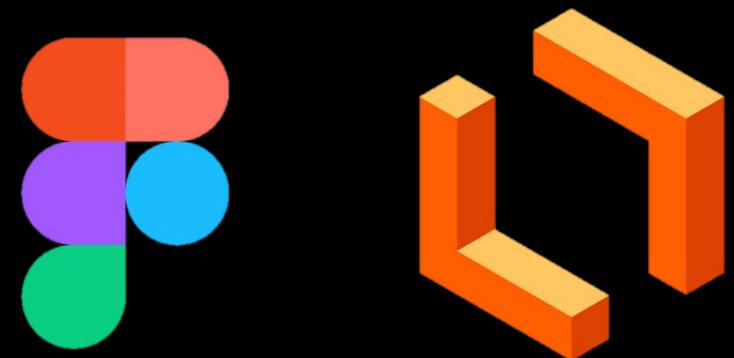


데이터 흐름의 주요 요소

- 데이터 입력: 사용자로부터 데이터 수집(예: 웹 폼)
- 데이터 처리: 변환, 계산, 필터링(예: 백엔드 로직)
- 데이터 저장 및 출력: 데이터베이스 저장 후 결과 반환



설계 단계에서의 협업





설계 단계에서의 모범 사례





설계 단계의 도전 과제

문제

요구사항 변경

커뮤니케이션 부족

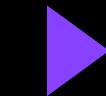
복잡한 설계

해결 방안

요구사항 변경 관리

정기적인 설계 리뷰

단순성 유지

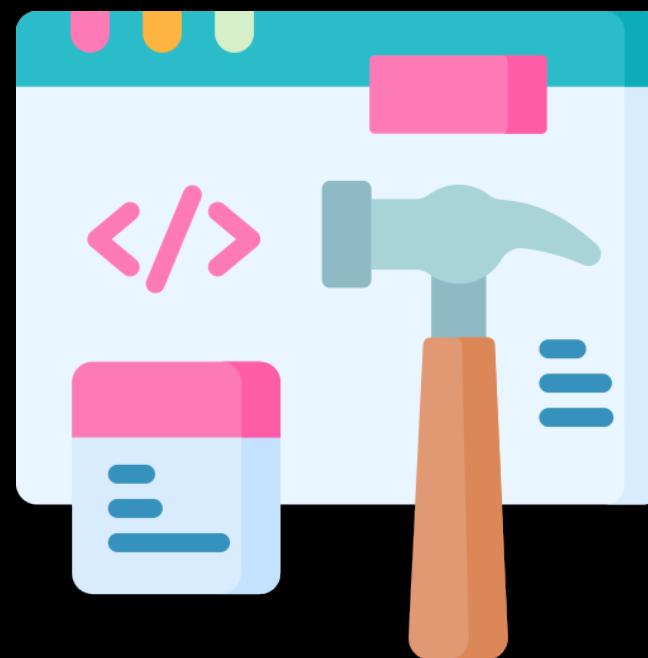




QUIZ 12번부터 13번을 풀어봐요!



설계를 구현으로 연결하기



설계 단계에서 구체화된 계획과 아키텍처를
실제 소프트웨어로 전환



구현 단계의 코딩 원칙

Clean Code의 주요원칙 (General)

- Follow Standard Conventions** Coding 표준, 아키텍처 표준 및 설계 가이드를 준수하라.
- Keep It Simple, Stupid (KISS)** 단순한 것이 효율적이며, 복잡함을 최소화하라.
- Boy Scout Rule** 캠핑장을 떠나기 전에 원래보다 깨끗하게 해야 한다.
(참조되거나 수정되는 코드는 원래보다 clean하게 해야 함)
- Root Cause Analysis** 항상 근본적인 원인을 찾아라. 그렇지 않으면 반복될 것이다
- Do Not Multiple Languages in One Source File** 하나의 파일은 하나의 언어로 작성하라. (Java, JavaScript, Html...)

출처 : Clean Code Cheat Sheet <https://www.planetgeek.ch/wp-content/uploads/2014/11/Clean-Code-V2.4.pdf>

코딩 원칙

클린 코드

DRY(Don't Repeat Yourself)

SOLID 원칙



소프트웨어 빌드의 개념

[빌드 프로세스]

1. *의존성 설치

2. 소스 코드 *컴파일

4. 오류 및 경고 검토

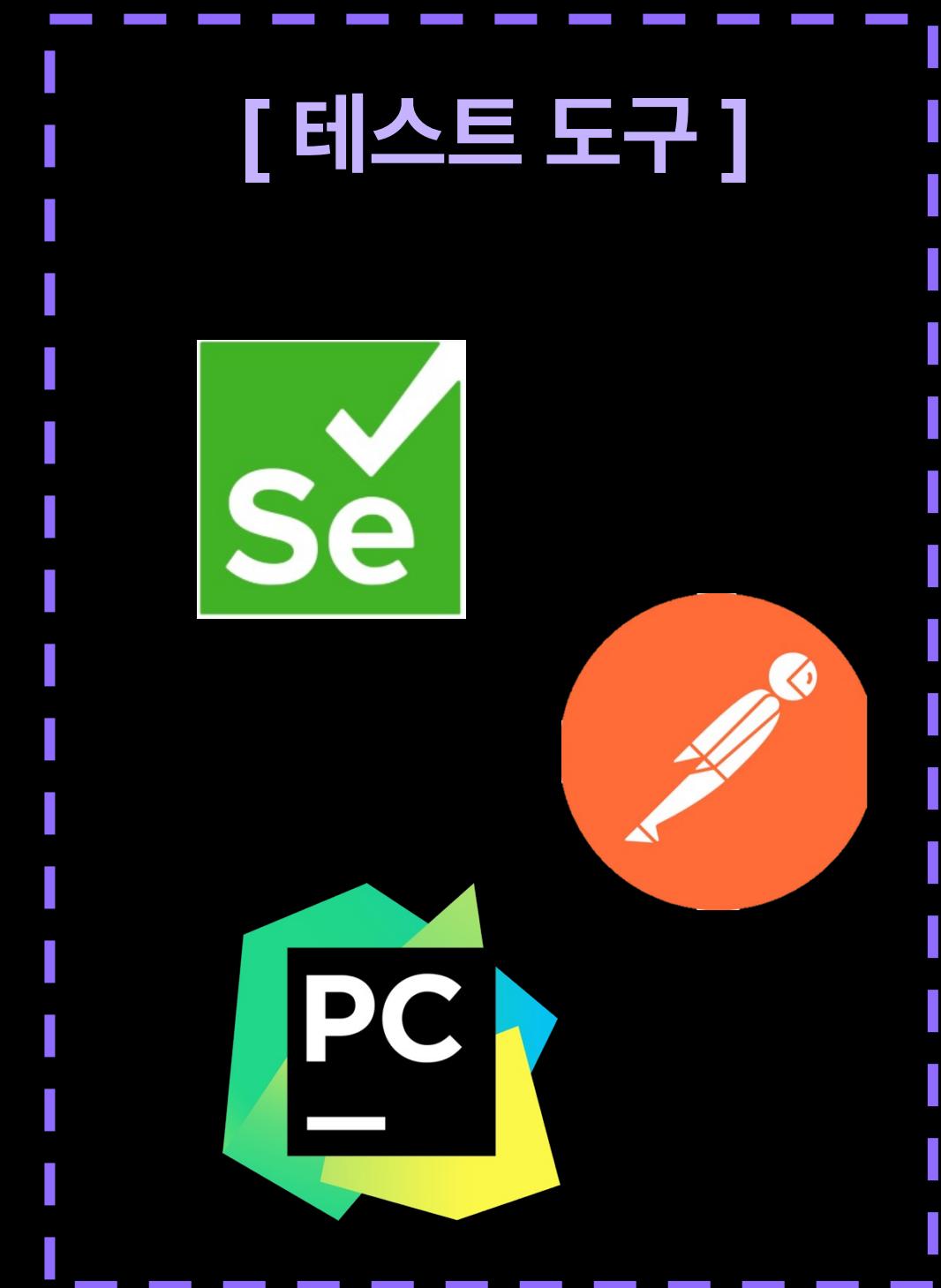
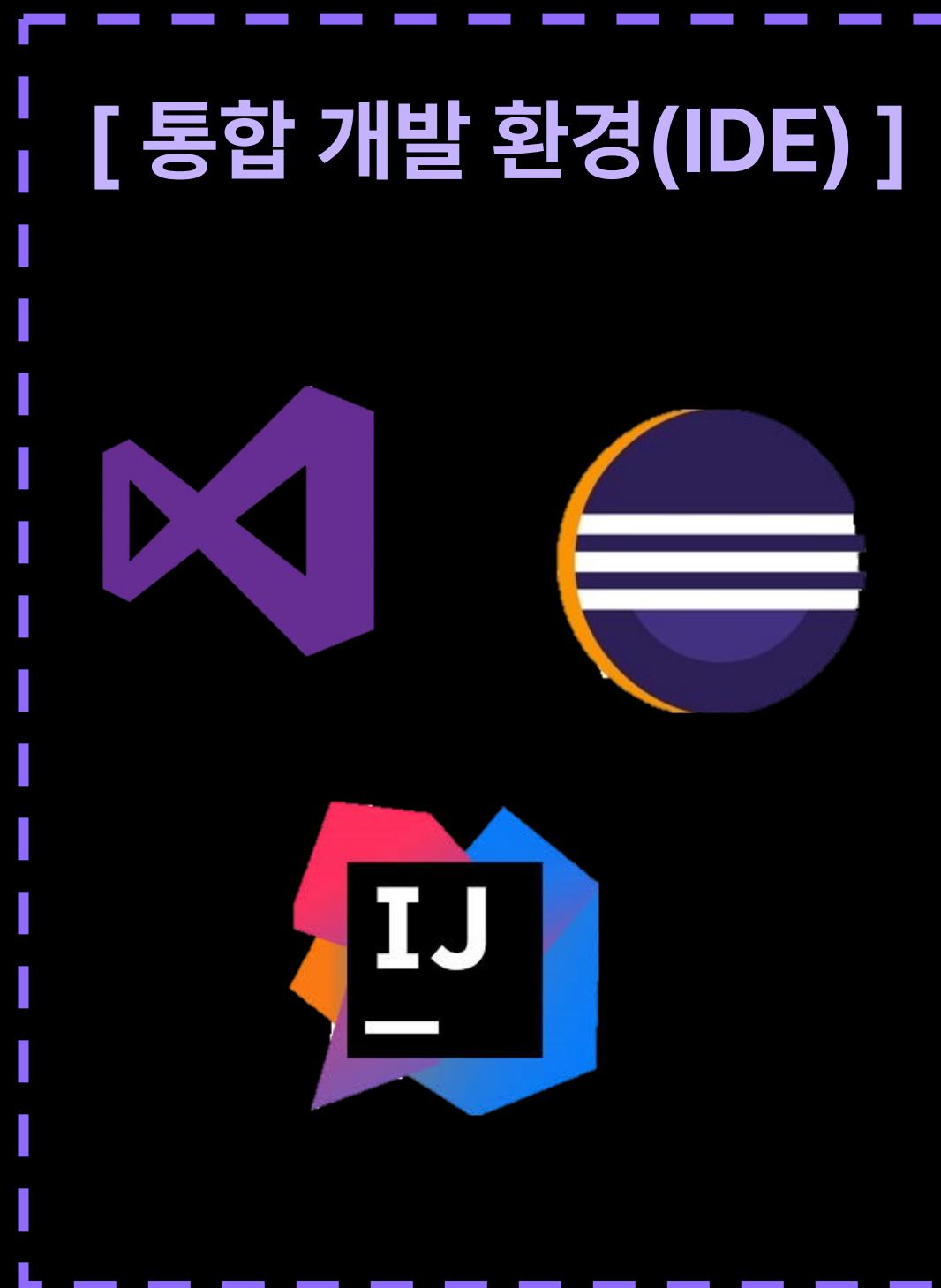
3. 실행 파일 생성

*컴파일 : 소스 코드를 사람이 읽을 수 있는 고급 프로그래밍 언어에서 기계가 이해할 수 있는 저급 언어(바이너리 코드 또는 기계어)로 변환하는 과정

*의존성: 소프트웨어가 실행되거나 빌드될 때 얼마나 외부 라이브러리, 모듈, 프레임워크 등에 종속되는지

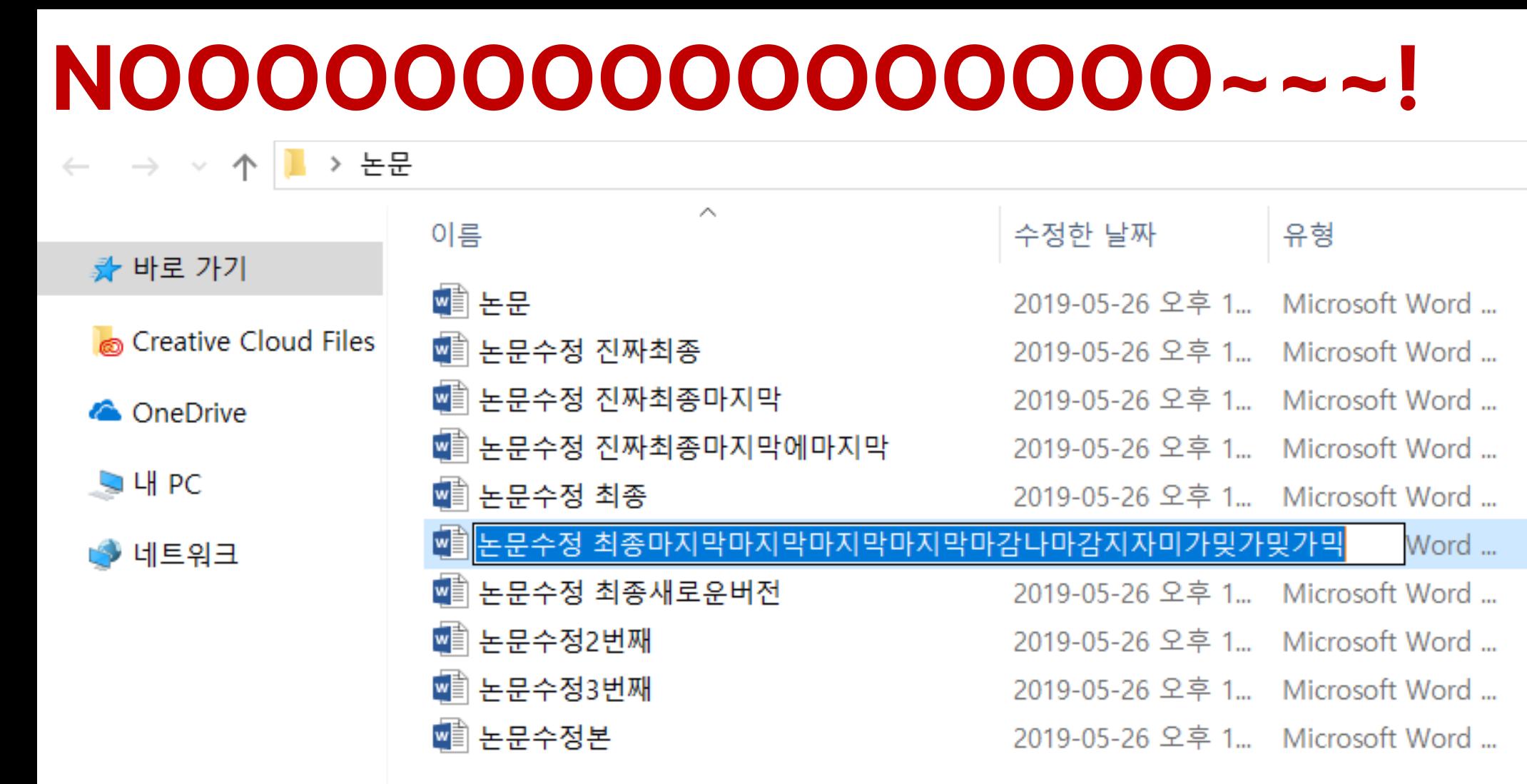


개발 도구의 활용





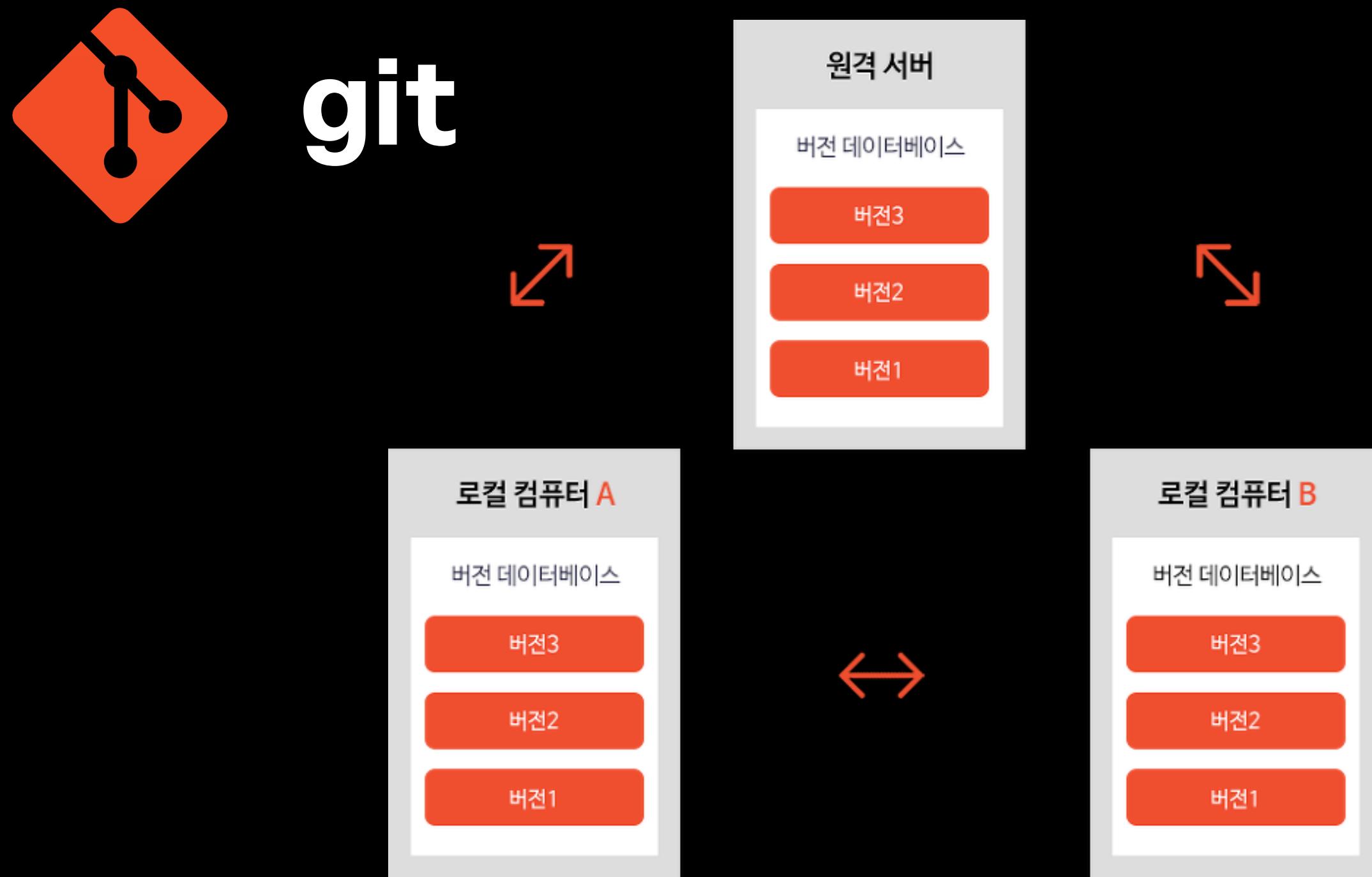
계속해서 업데이트되는 버전을 어떻게 관리하면 좋을까?





버전 관리의 개념

버전 관리 : 소스 코드의 변경 사항을 추적하고, 팀원 간 협업을 지원하는 시스템



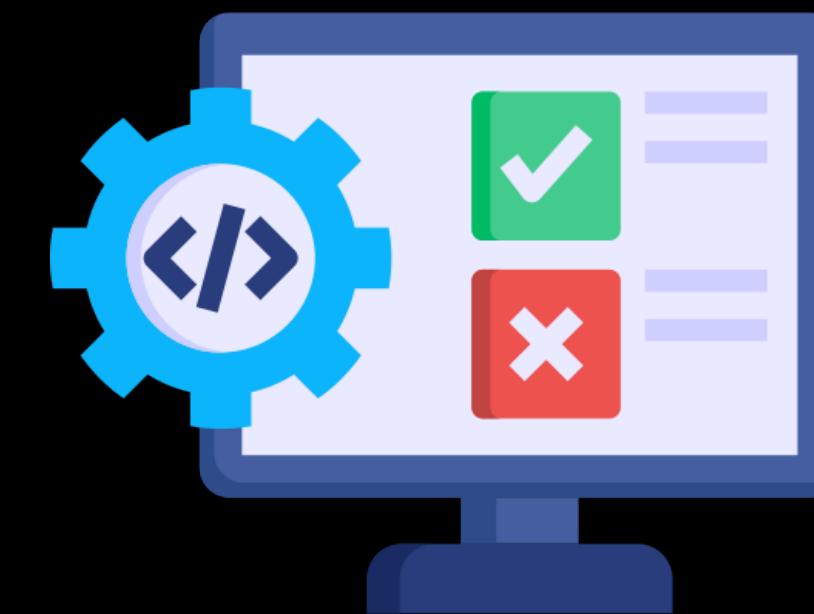


퀴즈 타임



QUIZ 14번부터 15번을 풀어봐요!

구현에서 테스트로, 품질 보장의 첫 걸음



구현 단계에서 개발된 소프트웨어가
요구사항을 충족하는지 **검증**



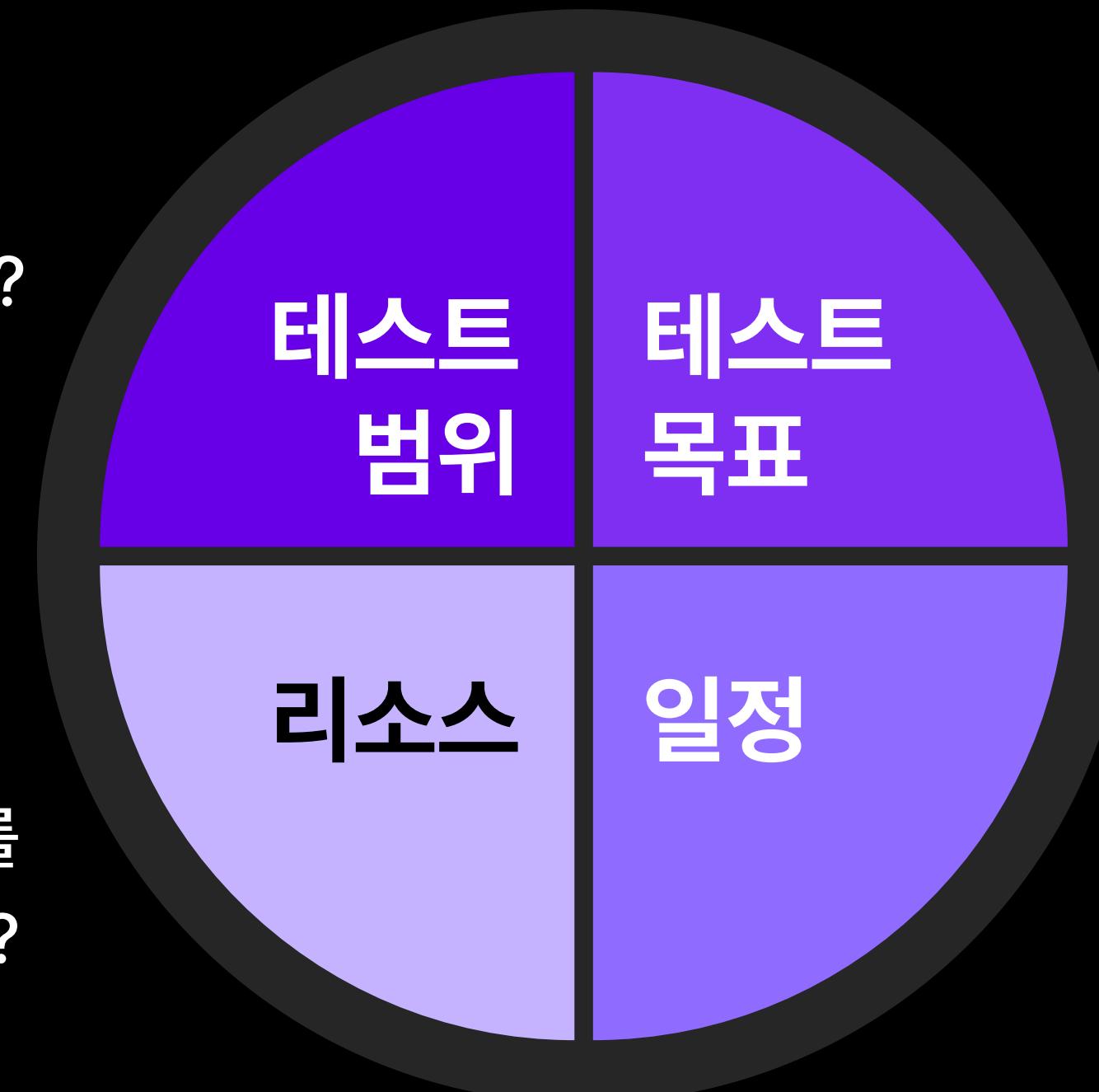
테스트 계획이란 무엇인가

[테스트 계획의 주요 구성 요소]

무엇을 테스트할 것인가?

어떤 문제를 발견할 것인가?

누가, 어떤 도구로 테스트를
수행할 것인가?



테스트 일정 및 마일스톤 설정



테스트 케이스, 테스트의 핵심

테스트



특정 조건에서 소프트웨어의 기능과 동작을 확인하기 위해
설계된 테스트 시나리오



어떤 테스트를 해야 할까요?

테스트 유형	목적	주요 활동	사용 예시
기능 테스트	소프트웨어가 요구된 기능을 제대로 수행하는지 검증	입력값과 기대 결과를 비교하여 정상 작동 여부 확인	로그인, 회원가입, 데이터 저장/검색 기능 테스트
성능 테스트	소프트웨어의 속도, 응답 시간, 처리량 등을 평가	부하 테스트(Load Test), 스트레스 테스트(Stress Test)	대량 트래픽 처리, 파일 업로드 속도 확인
회귀 테스트	새로운 코드 변경이 기존 기능에 문제를 발생시키지 않는지 확인	기존 테스트 케이스 재실행, 변경된 부분에 집중 테스트	버그 수정 후 주요 기능 테스트
보안 테스트	시스템의 보안 취약점을 탐지	침입 시도 시뮬레이션, 데이터 암호화 확인, 인증/권한 검증	비밀번호 암호화, 데이터 유출 방지 확인



테스트 자동화로 더 빠르게

테스트 자동화 : 반복적인 테스트를 자동으로 실행하기 위해 스크립트를 사용하는 방법

자동화 도구: Selenium, Cypress, JUnit, TestNG



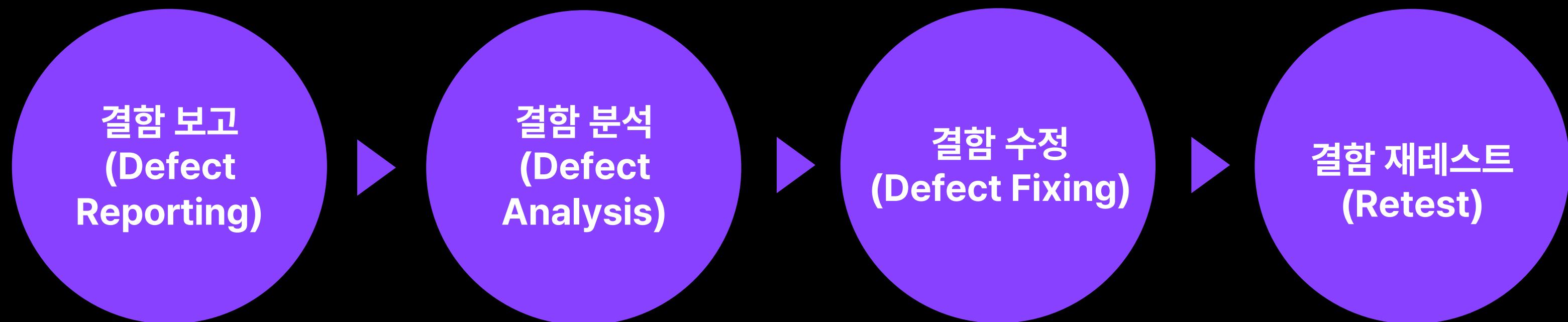
JUnit

TestNG

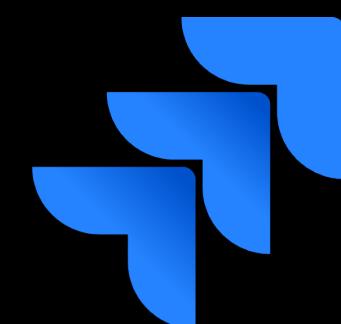


결함 발견하고 수정하기

[결함 관리 프로세스]



결함 관리 도구: Jira, Bugzilla, Trello





협업으로 테스트 효율성 극대화



1. 정기적인 테스트 리뷰 회의 개최
2. 테스트 결과를 공유하고, 결함 우선순위 결정
3. QA, 개발자, 이해관계자 간의 명확한 커뮤니케이션



테스트 결과를 명확히 전달하기

Test Report						
Test Cycle	System Test					
EXECUTED	PASSED			130		
	FAILED			0		
	(Total) TESTS EXECUTED (PASSED + FAILED)			130		
	PENDING			0		
IN PROGRESS				0		
BLOCKED				0		
(Sub-Total) TEST PLANNED (PENDING + IN PROGRESS + BLOCKED + TEST EXECUTED)				130		
Functions						
Functions	Description	% TCs Executed	% TCs Passed	TCs pending	Priority	Remarks
New Customer	Check new Customer is created	100%	100%	0	High	
Edit Customer	Check Customer can be edited	100%	100%	0	High	
New Account	Check New account is added	100%	100%	0	High	
Edit Account	Check Account is edit	100%	100%	0	High	
Delete Account	Verify Account is delete	100%	100%	0	High	
Delete customer	Verify Customer is Deleted	100%	100%	0	High	
Mini Statement	Verify Ministatement is generated	100%	100%	0	High	
Customized Statement	Check Customized Statement is generated	100%	100%	0	High	

[테스트 보고서 구성 요소]

- 테스트 목적과 범위
- 테스트 케이스 실행 결과
- 결함 요약 및 우선순위
- 결론 및 제안 사항



주요 테스트 도구 비교



Selenium: 웹 애플리케이션 자동화

JUnit

JUnit: Java 기반 단위 테스트



Postman: API 테스트 도구



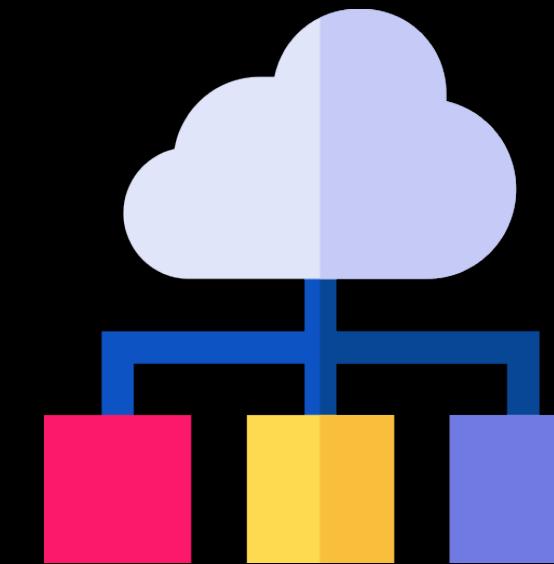
퀴즈 타임



QUIZ 16번부터 17번을 풀어봐요!



소프트웨어를 세상에 공개하기



테스트가 완료된 소프트웨어를 사용자에게 배포하고,
사용자 피드백을 통해 개선 및 안정화

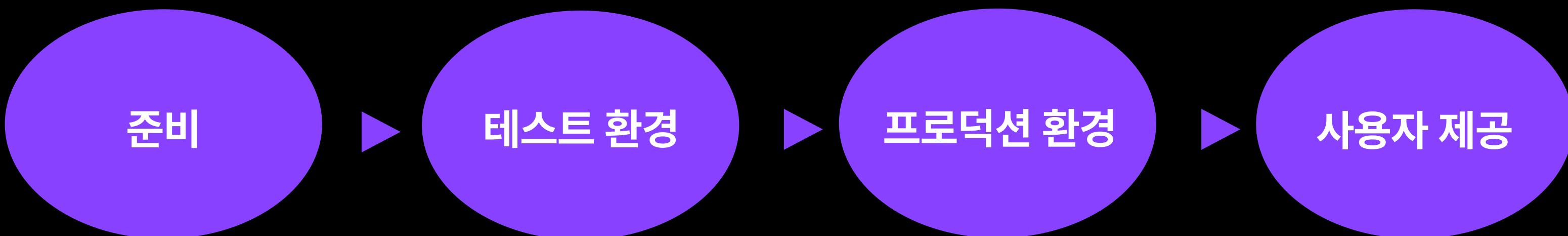


배포란 무엇인가?

배포

개발된 소프트웨어를 실제 환경에 배포하여 사용자에게 제공하는 과정

[배포 프로세스]

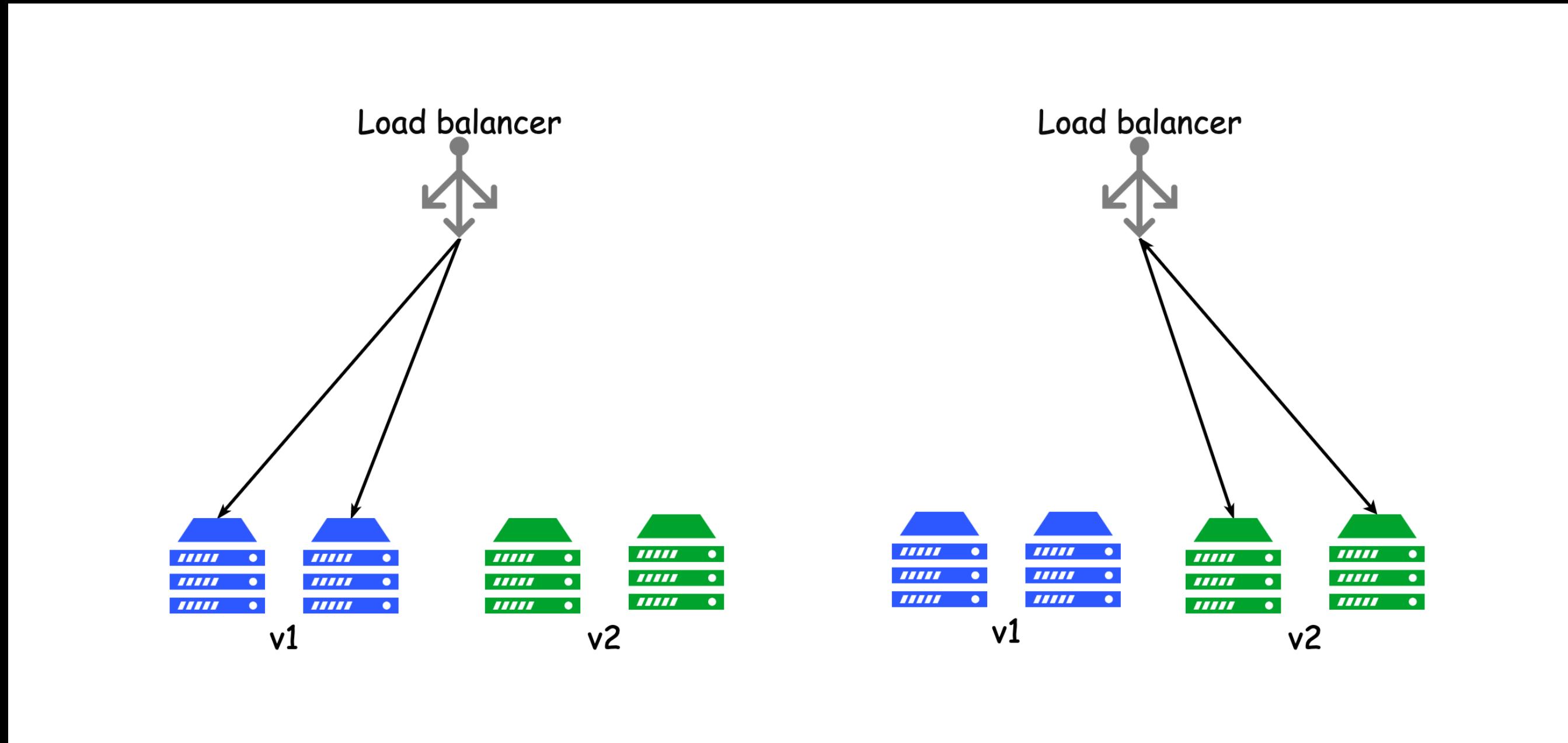




성공적인 배포를 위한 전략



Blue/Green : 새로운 버전과 기존 버전을 병렬로 운영하여 전환

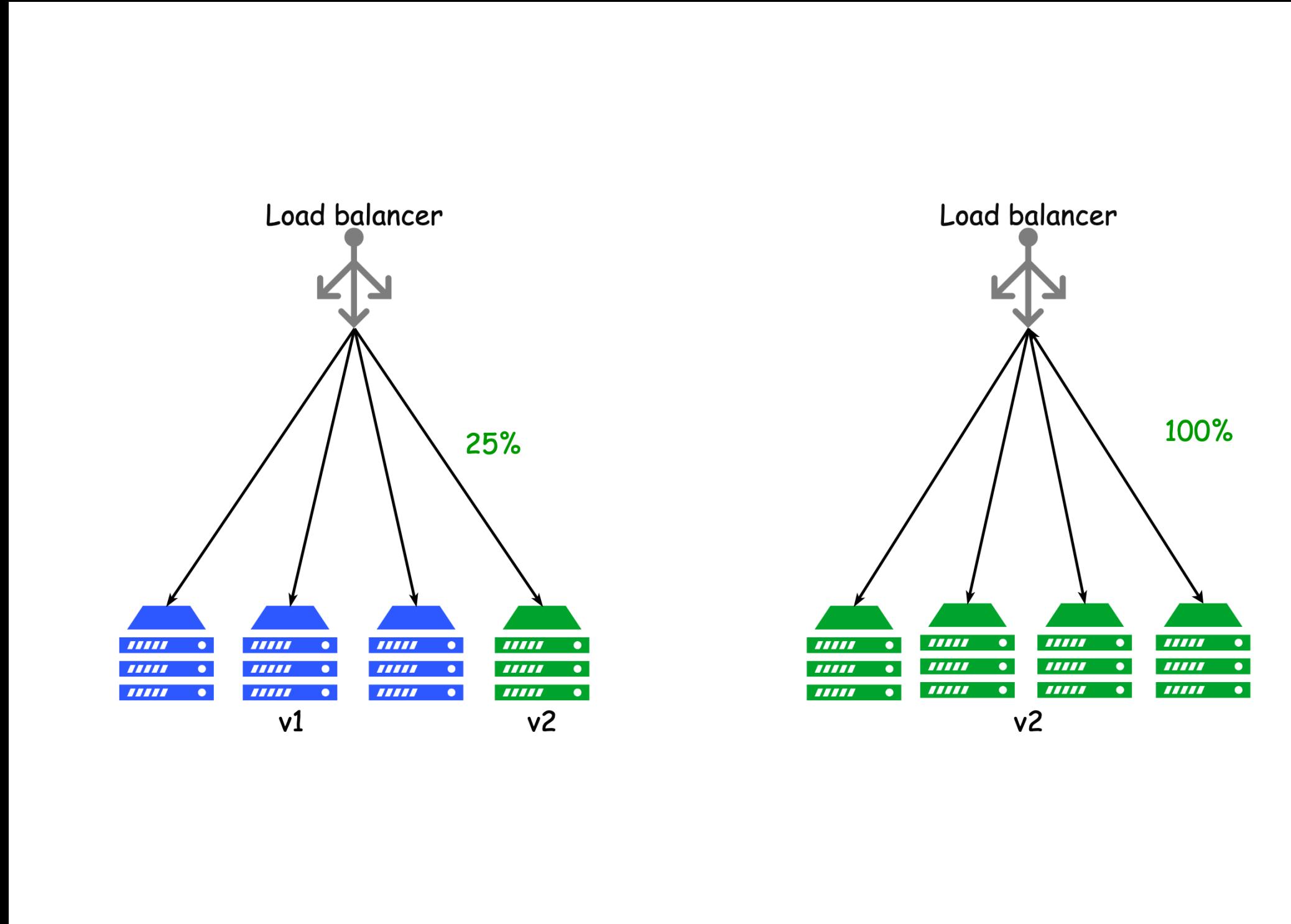




성공적인 배포를 위한 전략



Canary : 소규모 사용자 그룹에 먼저 배포 후 안정성을 확인

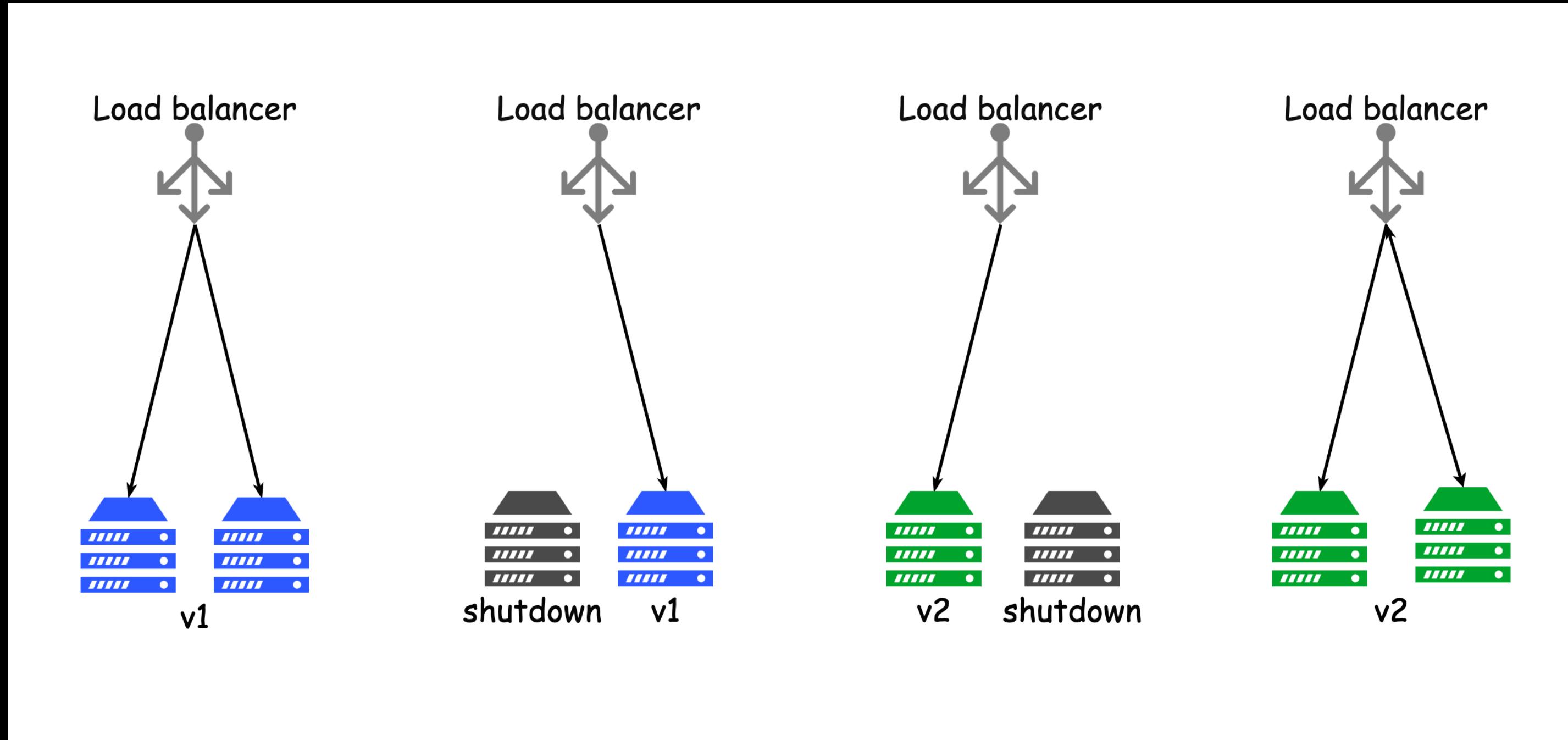




성공적인 배포를 위한 전략



Rolling : 일부 사용자에게 먼저 배포 후 점진적으로 확장

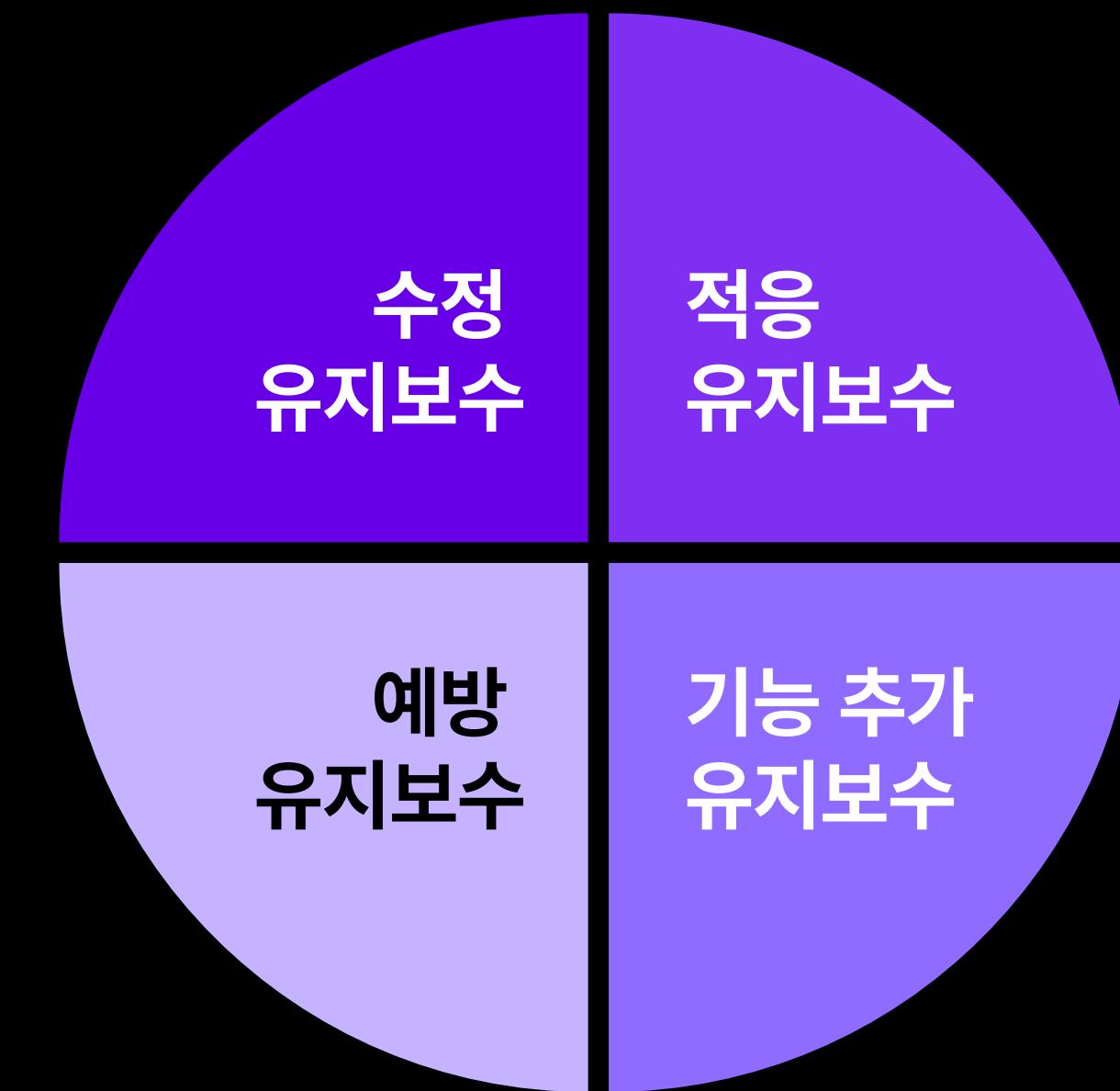




유지보수란 무엇인가?

유지보수

소프트웨어를 배포한 이후에도 안정적으로 작동하고 지속적으로 개선할 수 있도록 관리하는 과정



[유지보수의 유형]



사용자 피드백 수집



사용자 피드백

온라인 설문조사

애널리틱스 도구(Google Analytics, Firebase)

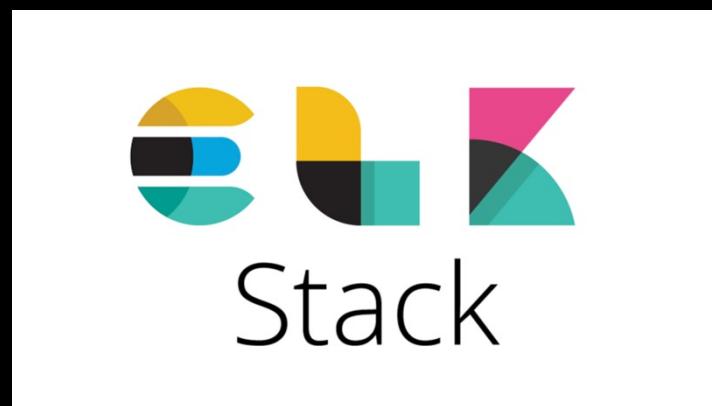
사용자 리뷰 및 평가(App Store, Play Store 리뷰)



유지보수를 더 빠르고 정확하게

자동화 도구

[Log 관리 도구]



Splunk, ELK Stack

[모니터링 도구]



New Relic, Datadog

[자동화 배포 도구]



Jenkins, GitLab CI/CD



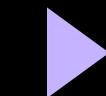
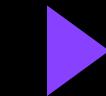
유지보수 과정에서의 도전

문제

사용자 피드백 부족

환경 변화

긴급 결함



해결 방안

사용자와 지속적 소통

실시간 모니터링

예방 유지보수 강화



유지보수를 성공적으로 관리하는 법

[핵심 원칙]

1. 사용자의 피드백을 기반으로 한 지속적 개선

2. 문서화를 통한 유지보수 과정의 체계화

3. 장기적인 안정성을 고려한 업데이트



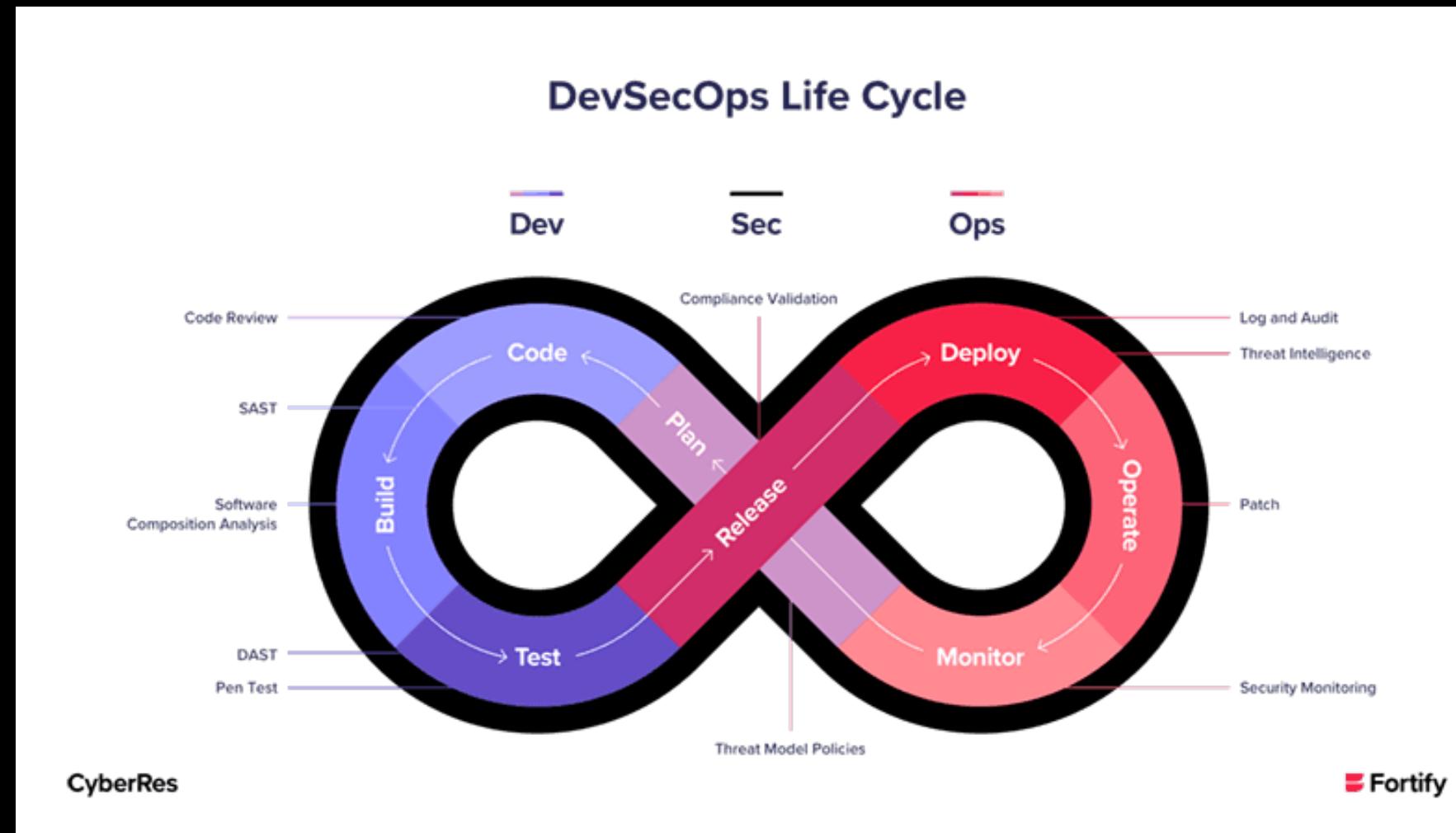
퀴즈 타임



QUIZ 18번부터 20번을 풀어봐요!



SDLC, 왜 필요한가?



소프트웨어 개발의 체계적 접근 방식을 제공하며,
소프트웨어 품질과 개발 효율성을 보장

장점과 한계에 대해 살펴보기



SDLC의 강력한 장점

[SDLC의 주요 장점]

효율성

품질 보장

비용 절감

명확한 문서화





하지만, SDLC에도 한계가 있다.

[SDLC의 주요 장점]

유연성 부족

시간과 비용 소모

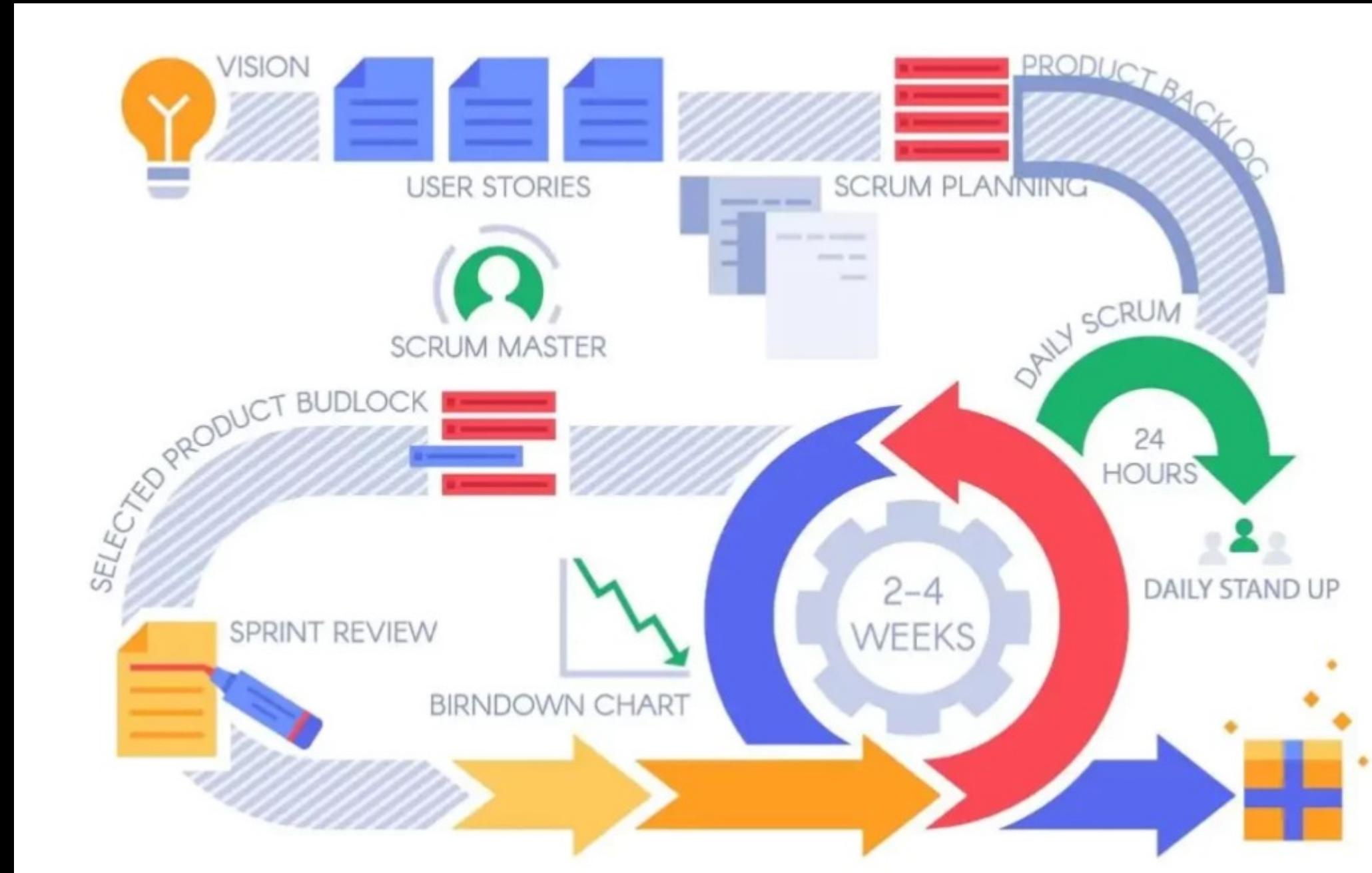
복잡성 증가

모델 선택의 어려움





SDLC의 한계를 극복하는 방법



1. 유연한 모델 사용
2. 초기 단계의 요구사항 강화



SDLC의 한계를 극복하는 방법



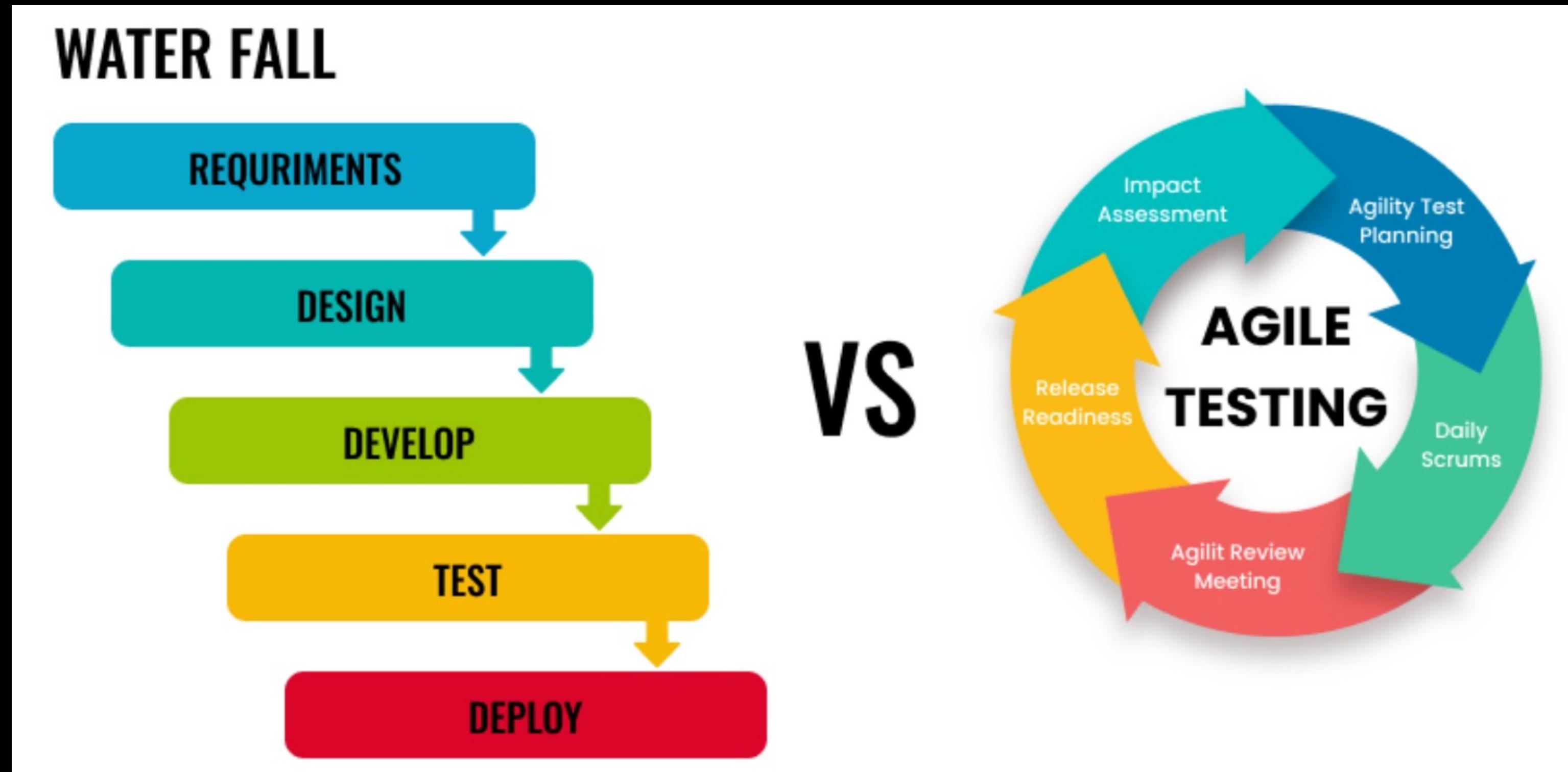
3. 효율적인 의사소통 도구 활용



4. 지속적 교육



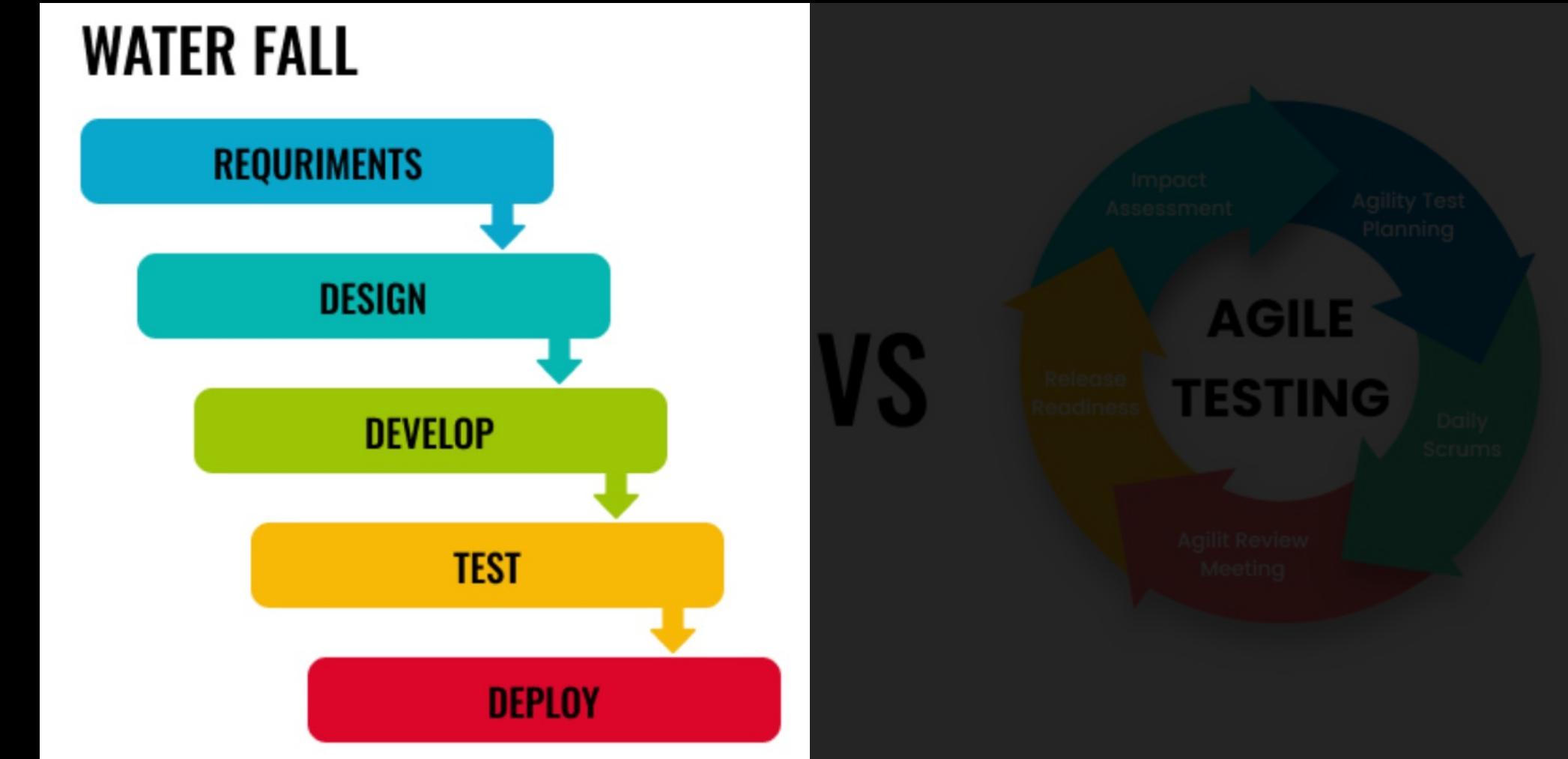
SDLC 모델 선택이 중요한 이유



프로젝트 특성에 따라 적합한 SDLC 모델을 선택하는 것은 성공의 핵심



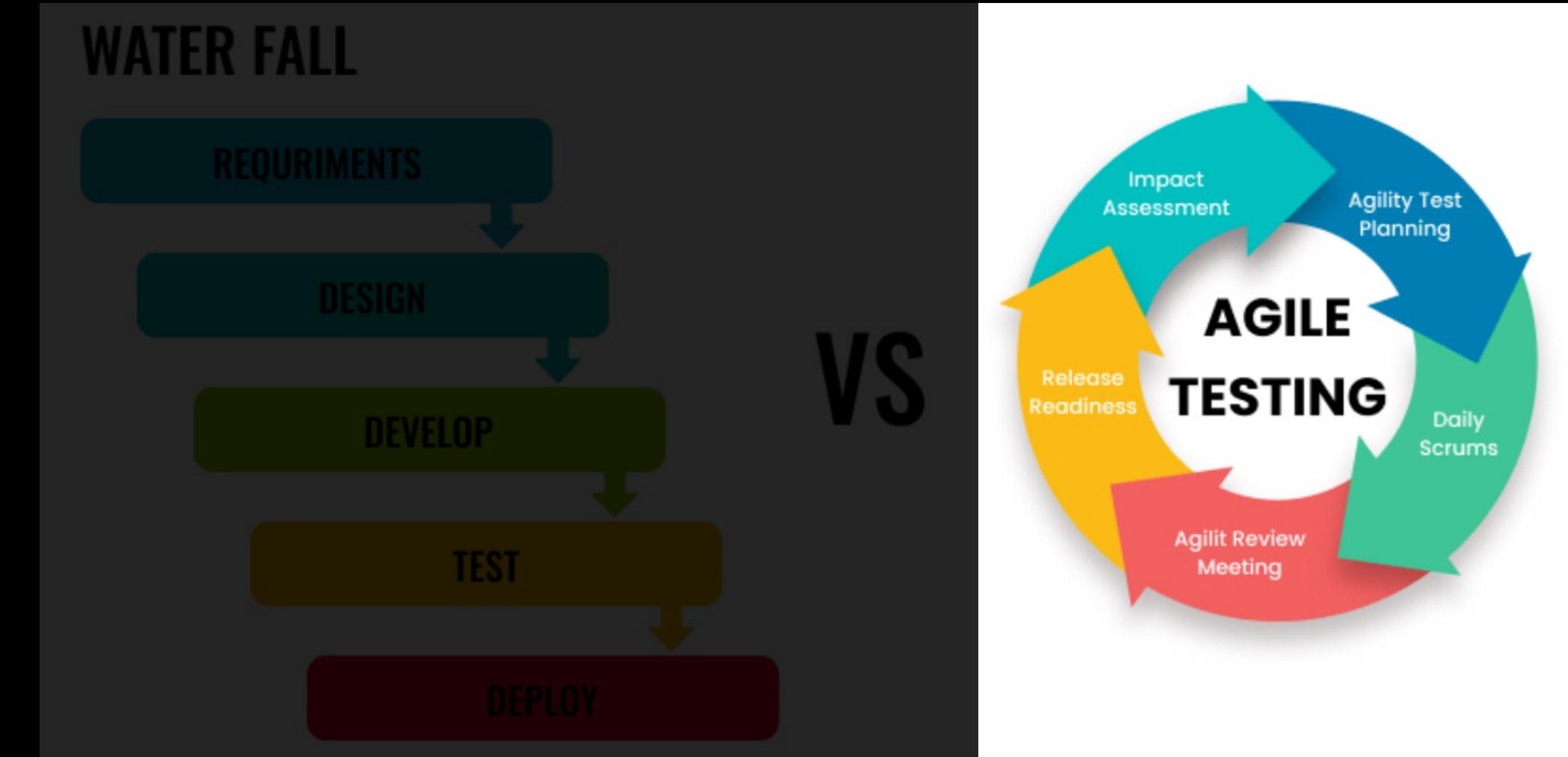
Waterfall 모델이란



정의	단계별 순차적으로 진행되는 소프트웨어 개발 모델
특징	<ol style="list-style-type: none">각 단계가 완료된 후 다음 단계로 진행명확한 문서화와 계획 기반단계 간 되돌아갈 수 없는 구조
적합한 프로젝트	요구사항이 명확하고, 변경 가능성이 적은 프로젝트



Agile 모델이란



정의	변화에 유연하게 대처하며, 반복적이고 점진적으로 개발을 진행하는 모델
특징	<ol style="list-style-type: none">지속적인 피드백과 개선스프린트를 통한 반복적 개발고객과의 긴밀한 협업
적합한 프로젝트	요구사항이 자주 변경되거나, 빠른 결과물이 필요한 프로젝트



Waterfall vs Agile

특징	Waterfall	Agile
진행 방식	순차적(한 단계 완료 후 다음 단계 진행)	반복적(스프린트를 통해 점진적 개발)
유연성	변경 사항 반영 어려움	변경 사항에 유연하게 대응 가능
문서화	철저한 문서화	필요한 경우에만 문서화
고객 참여	초기 요구사항 정의 후 제한적 참여	지속적인 협업과 피드백
적합한 프로젝트	요구사항이 명확한 프로젝트	변화 가능성이 높은 프로젝트



퀴즈 타임



QUIZ 21번부터 22번을 풀어봐요!