데이터 분석 기반
AI 시스템 개발자 양성 과정

02.소프트웨어

조창제

강의자료

2025.08.

01. 정의 및 특성

1.정의

① 컴퓨터 시스템을 제어하거나 특정 작업을 수행하기 위해 작성된 명령어 집합

② 실행 가능한 코드뿐 아니라, 메뉴얼, 설명서, 데이터 등 부속 자료까지 포함

2.특성

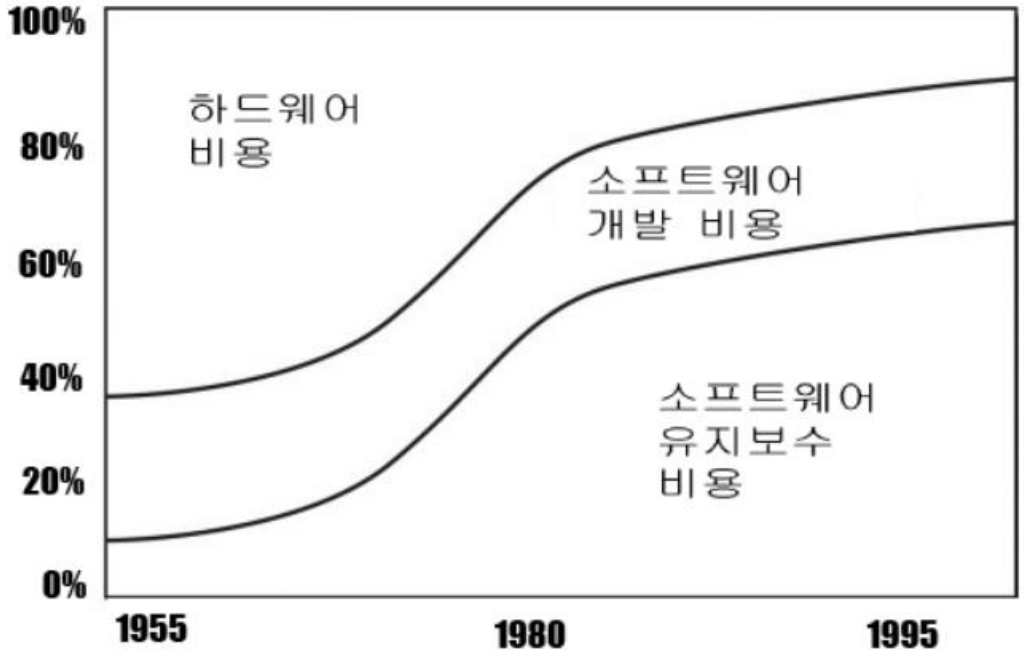
특성	설명
비가시성(Invisibility)	소프트웨어의 구조가 보이지 않으며, 내부에서 코드로 존재
복잡성(Complexity)	정형화된 구조가 없음
비마모성(Non-wearability)	물리적 마모가 존재하지 않음
복제성(Replicability)	빠르고 간단하게 복제 가능
변경성(Changeability)	필요에 따라 지속 수정 개선이 가능
순응성(Conformability)	사용자 요구 및 환경 변화에 적응하여 변경 가능
무형성(Intangibility)	물리적 형태가 없음, 기능적 크기로 유형화하여 관리
개발(Development)	설계와 구현 과정이 중심

01. 정의 및 특성

1) 소프트웨어의 위기

① 점점 소프트웨어 규모와 복잡도는 증가, SW에 대한 이해 및 관리가 부재

② SW의 품질은 저하되고 유지보수 비용이 증가



문제점
비용 초과
유지보수
재작업
일정 지연 (납기일 미준수)
성능 저하 (요구사항 불만족)
신뢰성 저하 (오작동)
요구사항과 공급 간의 불균형

위기 대응
공학적 접근 (SW 방법론)
표준화 (SW 및 DATA 표준화)
관리 도구 (형상관리)
품질보증

1.정의

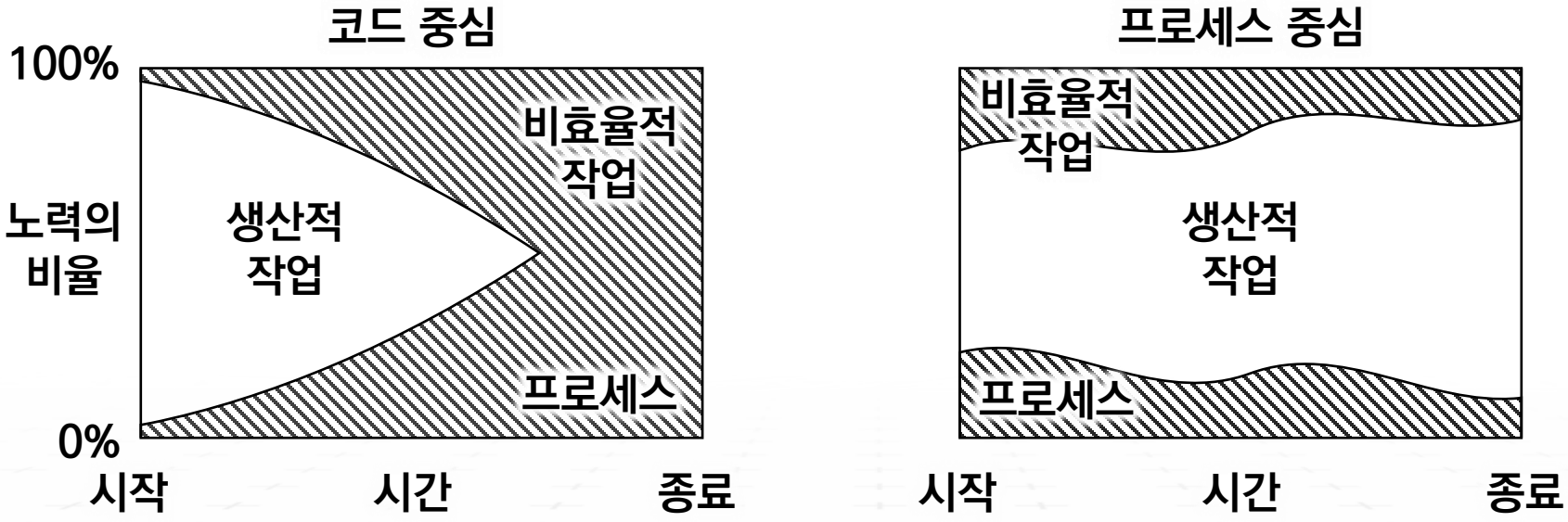
① 좋은 SW를 경제적으로 생산하기 위하여 연구하는 학문

1) 필요성

① SW는 점차 규모와 복잡도가 증가할 예정

② 프로세스에 관심을 두지 않고 개발에 중점 비효율 작업이 점차 늘어남

③ 체계적인 회사로 갈수록 소프트웨어 공학에 대한 이해도를 요구



2.소프트웨어 개발 생명주기(SDLC)

① SW의 개발의 타당성 검토부터 개발, 유지보수를 포함하는 전 과정을 하나의 생명주기로 단계별로 체계화한 모델을 의미

구성	설명
타당성 검토	사용자 요구사항을 정의하고 내용의 타당성을 검토, 진행여부 결정 수행 범위, 일정계획, ROI 산정
요구사항 분석	요구사항 상세화 기능 요구사항, 비기능 요구사항
소프트웨어 설계	요구사항 명세를 준수하는 SW 설계
소프트웨어 개발	요구사항을 준수하도록 코드 작성
소프트웨어 테스트	개발된 소프트웨어 검증 단위, 통합 테스트 등
운영 및 유지보수	문제점 수정, 기능 추가 등 인수 완료 후 발생하는 모든 개발 활동
소프트웨어 폐기	용도 변경, 사업철수 등으로 수명이 다한 SW 폐기 단종, 폐기

2.소프트웨어 개발 생명주기(SDLC)

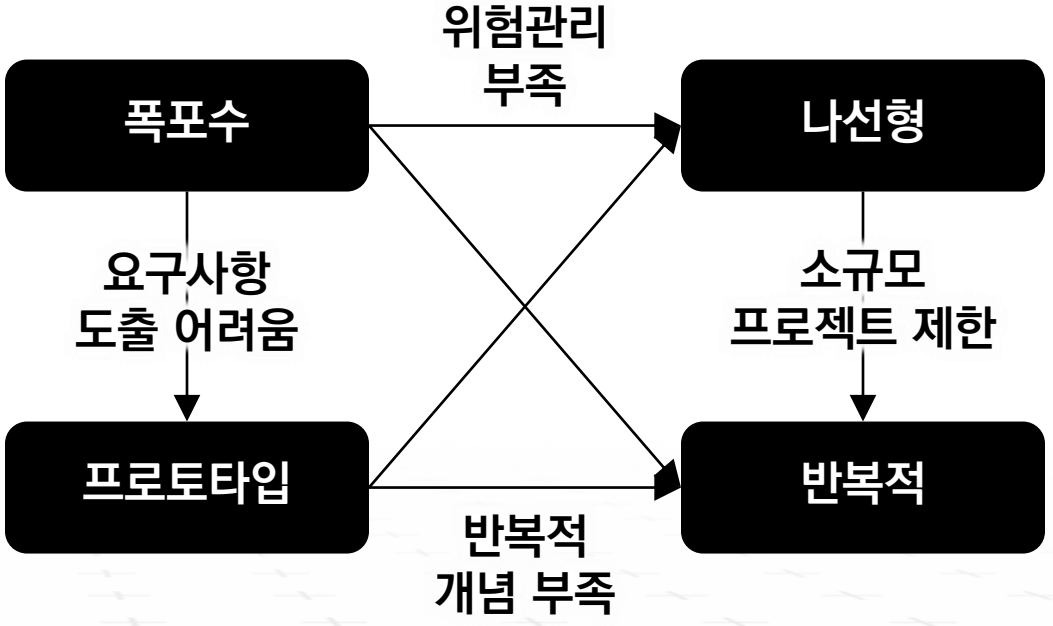
① SW의 개발의 타당성 검토부터 개발, 유지보수를 포함하는 전 과정을 하나의 생명주기로 단계별로 체계화한 모델을 의미

1) SW 개발 모델

① SW를 체계적이고 효율적으로 개발하기 위해 사용하는 절차와 구조화된 방식(**전반적 흐름**)

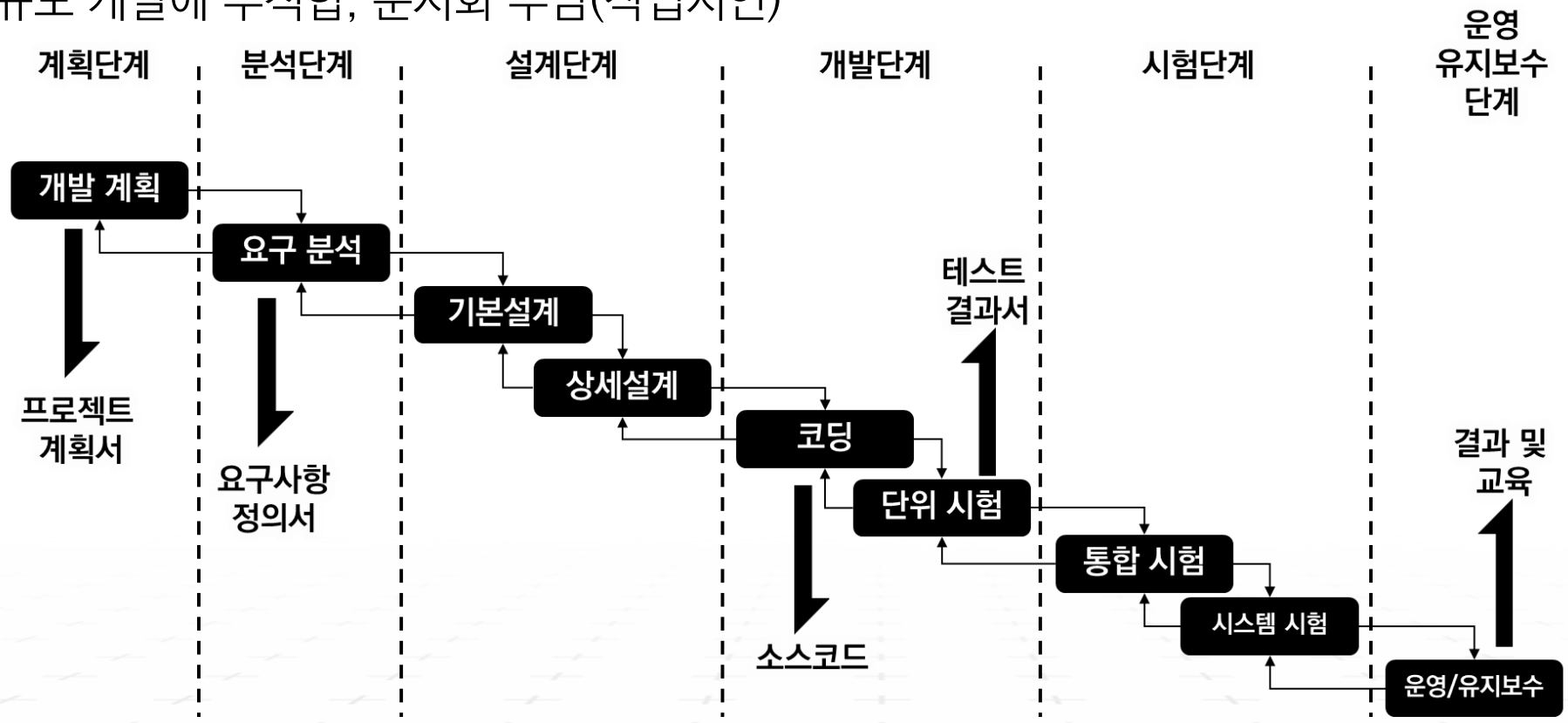
② 아래 예시 외에도 클린룸 모델, v 모델, 정형 명세 모델 등 여러가지 모델들이 존재함

모델	설명
폭포수 모델 (Waterfall)	요구사항 -> 설계 -> 구현 -> 테스트 -> 유지보수 단계별 순차 진행
프로토타입 모델 (Prototype)	초기 프로토타입 개발하여 사용자 피드백 반영 후 최종 개발
나선형 모델 (Spiral)	위험 분석을 기반으로 여러 번 반복하여 개발 진행
반복적 모델 (Iterative)	한 번에 개발하지 않고, 반복적으로 개선하는 방식



1.폭포수 모델

① 분석, 설계, 개발, 구현, 시험, 유지보수 단계를 순차적으로 접근하는 방법
② 적용 사례가 다수(낮은 위험성)
③ 대규모 개발에 부적합, 문서화 부담(작업지연)



1.개발 방법론

① SW를 체계적으로 개발하기 위한 방법론(**어떻게 실행할지에 대한 전략**)
② 효율성, 품질, 유지보수성, 일관성을 보장하는 것이 중요
③ 현대에 와서는 DevOps로 발전

구조적 방법 → **정보공학 방법** → **객체지향 방법** → **CBD 방법**

1970년대
절차/기능 중심

1980년대
자료구조 중심

1990년대
객체 중심
(프로세스 + 데이터)

2000년대
컴포넌트 중심

Waterfall → **ASD** → **RAD, Scrum XP** → **Agile**

단계별 순차적 진행

반복적인 개발
적응성 강조

RAD: 빠른 개발 주기 강조
Scrum: 1~4주 단위 반복 개발
자율성 협업 강조
XP: TDD, Pair Programming, CI

애자일 선언
12가지 원칙

1.객체지향 방법론

① 개체(Entity)를 속성(Attribute)과 메소드(Method)가 결합된 형태의 객체(Object)로 표현하는 개념
② 효율성, 품질, 유지보수성, 일관성을 보장하는 것이 중요

구성 요소	설명
객체	데이터(속성)와 동작(메소드)을 포함하는 독립 단위
클래스	객체를 정의하는 틀
메시지	객체 간의 상호 작용(메소드 호출)
메소드	객체가 수행하는 동작(함수)
속성	객체가 가지는 데이터(변수)

캡슐화	추상화	다형성	정보은닉	상속
객체 내부 데이터 변경을 제한	공통된 기능을 일반화 유지보수성 향상	코드 재사용성 유지보수 용이	불필요한 접근을 차단	부모 클래스의 기능을 상속

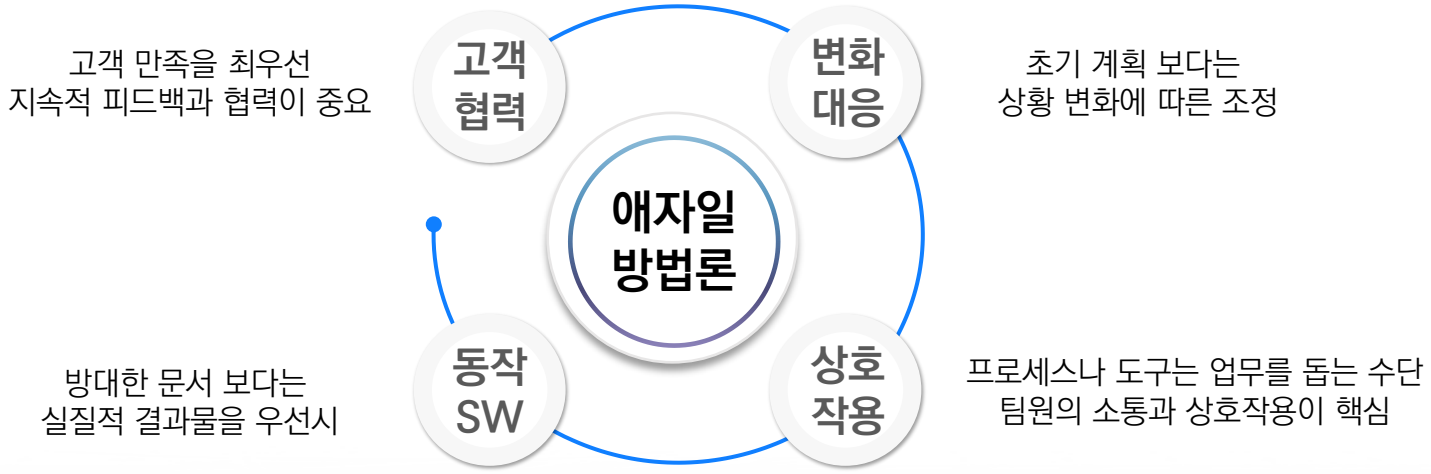
1.컴포넌트 기반 방법론

① 객체지향의 단점인 S/W 재사용성을 극대화한 개발 방법론, 기 개발된 컴포넌트를 활용
② 효율성, 품질, 유지보수성, 일관성을 보장하는 것이 중요

구성 요소	설명
컴포넌트	독립적으로 실행가능한 SW 모듈
인터페이스	컴포넌트 간의 통신을 위한 표준 인터페이스
프레임워크	컴포넌트를 배포/실행/관리 하는 환경
미들웨어	컴포넌트 간의 연동을 지원하는 SW 계층
리포지토리	재사용 가능한 컴포넌트를 저장하고 관리하는 공간

2.애자일 방법론

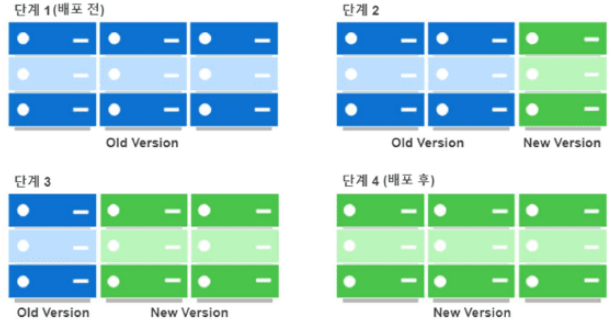
① 짧은 개발 주기로 기능을 반복적으로 개발 및 배포하여 빠르게 피드백을 반영하는 SW 개발 방법론
② 반복적 개발, 짧은 주기, 빠른 피드백, 협업 강화, 변화 대응
③ 대표적 프레임워크로 스크럼(Scrum), 칸반(Kanban), XP(Extreme Programming), Lean 존재



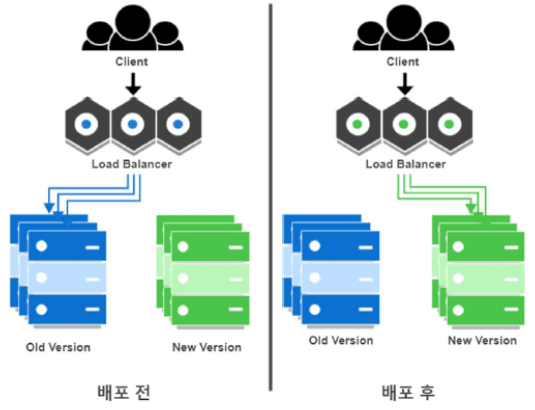
3.CI/CD(무중단 배포)

① 객체지향의 단점인 S/W 재사용성을 극대화한 개발 방법론, 기 개발된 컴포넌트를 활용
② 효율성, 품질, 유지보수성, 일관성을 보장하는 것이 중요

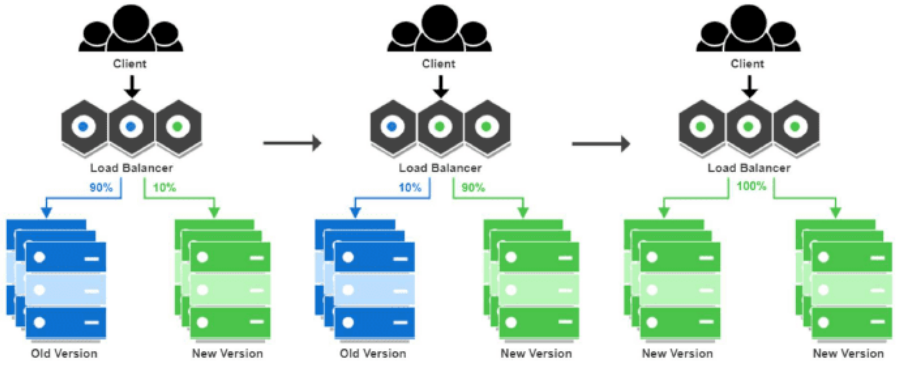
구성 요소	설명
롤링 배포	점진적으로 서버를 교체하며 배포
블루그린	두 개의 환경을 운영하며 트래픽을 전환
카나리아 배포	일부 사용자에게 먼저 배포 후 점진적으로 확장
기능 플래그 배포	기능을 배포해두고 특정 시점에 활성화
빅뱅 배포	한번에 전체 시스템을 새로운 버전으로 변경



[그림 2] 롤링 배포



[그림 3] 블루-그린 배포



[그림 4] 카나리 배포

4.마이크로서비스 아키텍처

1) 개념

① 하나의 큰 어플리케이션을 여러 개의 작고 독립적인 서비스로 쪼개어 개발/배포/운영 하는 아키텍처 스타일

② 서비스 단위 분리

- 회원 서비스, 상품 서비스, 결제 서비스, 주문 서비스 등 으로 세분화해서 나눔

③ 독립 배포/확장

- 특정 서비스만 수정/배포 가능
  - 필요한 서비스만 스케일 확장 가능

2) 장/단점

① 빠른 배포, 변경이 용이(CI/CD에 적합)

② 유연한 확장성, 조직 분리 용이

③ 데이터 일관성 관리 어려움

④ 배포/운영 인프라 관리가 복잡해짐

1.요구공학

① 시스템의 개발, 변경의 목적을 식별하기 위해 이해관계자들의 요구를 이해 및 조정하여 체계적으로 수집, 분석, 명세화, 검증하는 공정/학문

2.요구사항명세

3.소프트웨어 평가

① 국제 표준 존재
② ISO/IEC 9126, 14598, 12119, 25000 등 존재
③ 자동차, 의료기기에 대한 SW 안전 규격 별도 존재

4.소프트웨어 테스트

Thank You

Email: qkdrk777777@naver.com