**소프트웨어 개발수명주기**

\* 상위 수준에서 소프트웨어 개발 프로세스를 추상화해서 표현.

\* 개발 프로세스의 여러 단계와 활동 유형이 논리적, 시간상으로 서로 어떻게 연관되는지를 정의

\* 순차적 개발 모델, 반복적 개발 모델, 점진적 개발 모델 등이 있음.

\* 소프트웨어 개발 프로세스 내 일부 활동을 구체적인 소프트웨어 개발 방법과 애자일 실천법으로 설명

\* 인수 테스트 주도 개발, 행위 주고 개발, 도메인 주도 설계, 익스트림 프로그래밍, 기능 주도 개발, 칸반, 린,

IT 스크럼, 테스트 주도 개발 등이 있음.

**1. 소프트웨어 개발수명주기가 테스팅에 미치는 영향**

**\* 테스트 활동 범위 및 시기**

**\* 테스트 문서 상세화 수준**

**\* 테스트 기법 및 테스트 접근법 선택**

**\* 테스트 자동화 범위**

**\* 테스터의 역할과 책임**

**1.1 순차적 개발 모델**

**\* 테스터는 초기 단계에서 요구사항 리뷰와 테스트 분석 및 설계에 참여**

**\* 실행 가능한 코드는 주로 개발 후반에 생성되므로 동적 테스팅을 초기에 수행 불가**

**1.2 반복적 점진적 개발 모델**

**\* 반복 주기마다 프로토타입이나 제품을 증분이 생성**

**\* 각 반복 주기에서 모든 테스트 레벨에서 동적 및 정적 테스팅을 할 수 있으며,**

**빠른 피드백과 광범위한 리그레션 테스팅이 필요**

**1.3 에자일 소프트웨어 개발**

**\* 에자일에서는 프로젝트의 어느 시점에도 변화가 발생**

**\* 리그레션 테스팅을 용이하게 하기 위해 가벼운 작업 산출물과 테스트 자동화**

**\* 수동 테스팅은 사전 분석이나 설계가 필요 없는 경험 기반 테스트 기법으로 진행**

**2. 소프트웨어 개발 수명 주기와 우수한 테스팅 프랙티스**

\* 모든 소프트웨어 개발 활동에 상응하는 테스트 활동을 두어 모든 개발 활동이 품질 제어의 대상

\* 테스트 레벨마다 구체적이면서 독립적인 테스트 목적을 설정해 중복은 피하고 적절하면서 포괄적인 테스팅

\* 특정 테스트 레벨을 위한 테스트 분석과 설계를 소프트웨어 개발수명주기의 상응하는 각 개발 단계에서

시작해, 조기 테스팅 원칙을 준수.

\* 테스터가 문서 초안이 가용한 즉시 작업 산출물 리뷰에 참여하도록 해서 시프트-레이프 전략 지원을 위한

조기 테스팅과 결함 발견이 가능.

**3. 소프트웨어 개발 주도를 위한 테스팅**

\* 테스트 주도 개발 인수테스트 주도 개발 행위주도 개발은 서로 유사한 개발 접근법으로 개발 방향 결정을

위한 수단으로 테스트를 정의

\* 조기 테스팅 원리를 구현하고 시프트-레프트 접근법을 따르게 합니다. 반복적 개발 모델을 지원.

**3.1 테스트 주도 개발**

\* 테스트 케이스를 통해 코딩주도

\* 테스트를 먼저 작성하고 이를 충족하도록 코드를 작성한 다음 테스트와 코드를 리팩토링

**3.2 인수 테스트 주도 개발**

\* 시스템 설계 프로세스 중 인수 조건에서 테스트 도출

\* 테스트는 해당 테스트를 만족해야 할 애플리케이션 영역을 개발하기 전에 작성

**3.3 행위 주도 개발**

\* 애플리케이션의 기대 동작을 이해관계자가 이해하기 쉽도록 간단한 자연어로 작성해 테스트

케이스로 표현

\* 테스트 케이스는 자동으로 실행 가능한 테스트로 변환

\* 향후 적용/리팩토링 시 코드 품질 보장을 위해 자동화 테스트로 유지 가능

**4. 데브옵스와 테스팅**

\* 개발과 운영이 협력해 공통된 목표를 달성하도록 시너지 창출을 목표로 하는 조직 차원의 접근법

\* 개발과 운영이 가진 생각의 차이를 줄임과 동시에 각자 하는 일의 가치를 서로 동등 하게 보도록 조직문화의 변화가 필요

\* 팀의 자율성. 빠른 피드백, 통합 도구 체인, 지속적 통합과 지속적 배포와 같은 기술 실천법을 장려

\* 배포 파이프라인을 통해 높은 품질의 코드를 더 빠르게 빌드/테스트/릴리스 할 수 있음

**4.1 데브옵스 이점**

\* 빠른 피드백: 코드 품질과 변경 사항이 기존 코드에 미치는 영향을 신속하게 확인할 수 있음

\* 지속적인 통합: 개발자가 컴포넌트 테스트 및 정적 분석을 통해 높은 품질의 코드를 제출하도

록 유도하며, 시프트-레프트테스팅 접근법을 촉진

\* 안정적인 테스트 환경: 지속적 통합 및 배포 같은 자동화 프로세스가 안정적읜 테스트 구현

\* 자동화 테스트: 반복적인 수동 테스팅의 필요성을 줄이고 자동 리그레션 테스트의 규모와

범위를 확대하여 리그레션 발생 리스크를 최소화

**4.2 데브옵스 단점**

\* 배포 파이프라인 정의 및 설정 필요

\* 지속적 통합/배포 도구의 도입 및 유지 보수

\* 테스트 자동화를 위한 추가 지원 필요, 설정 및 유지보수가 어려움

**5.시프트-레프트 접근법**

\* 소프트웨어 개발수명주기 초기에 수행하도록 하는 접근법이기에 시프트-레프트라고 지칭

\* 시프트-레프트는 테스트를 더 일찍 수행해야 한다는 것을 의미하지만 그렇다고 소프트웨어 개발수명주기 후반 테스트를 무시해도 된다는 의미는 아님

\* 테스팅에서 시프트-레프트를 달성하는 좋은 프랙티스가 있으며 다음은 그중 일부를 나열

\* 테스팅 관점에서 명세를 리뷰한다 이런 명세 리뷰 활동을 통해 모호성 불완전성 불일치 등 잠재적인 결함을

발견하는 경우가 있음.

\* 코드를 작성하기 전에 테스트 케이스를 작성하고 코드 구현 중 코드를 테스트 하네스에서 실행

\* 빠른 피드백을 제공하고 코드 저장소에 소스 코드를 저장할 때 자동 컴포넌트 테스트를 함께 제출하도록 하는

지속적인 통합 가능하다면 지속적인 배포까지 적용한다

\* 동적 테스팅 전 또는 자동화된 프로세스의 일부로 소스 코드의 정적 분석을 완료

\* 가능한 한 컴포넌트 테스트에서부터 비기능 테스팅을 수행

\* 비기능 테스트는 완성 시스템과 실제 환경을 대변하는 테스트 환경이 가용한 소프트웨어 개발 수명주기 후반에 수행하는 경향이 있으므로 이는 일종의 시프트-레프트가 됨

\* 초기에 훈련 공수 비용이 추가로 들지만 프로세스 후반의 공수와 비용의 절감을 기대할 수 있음.

\* 시프트-레프트 접근법을 위해서는 이해관계자들이 개념을 이해하고 받아들이는 것이 중요.

**6. 회고 및 프로세스 개선**

\* 회고는 프로젝트나 반복 주기가 끝날 때 릴리스 마일 스톤에서 또는 필요시 진행

\* 회고의 시기와 구성은 사용 중인 소프트웨어 개발수명주기 모델에 따라 달라짐.

\* 회의에서 참가자는 다음에 대해 논의

**\* 무엇이 성공적이고 유지해야 할 것은 무엇인가?**

**\* 무엇이 부족했고 개선할 수 있는 점은 무엇인가?**

**\* 향후 개선 사항을 도입하고 성공 요소를 유지하려면 어떻게 해야 하는가?**

\* 결과는 기록해야 하며 이를 테스트 완료 보고서에 포함하는 경우가 많다 회고는 지속적인 개선을 성공적으로 구현하기 위해 반드시 필요하며 권장된 모든 개선 사항에 대한 후속 조치가 이루어지는 것이 중요

**6.1 회고의 이점**

\* **지속적인 개선**: 프로젝트나 반복 주기에서의 성공과 실패를 분석하여 향후 개선 사항을 도출

\* **테스트 효과성/효율성 향상**: 회고를 통해 테스트 프로세스를 개선하여 전반적인 테스트 품질

\* **테스트웨어 품질 향상**: 피드백을 통해 테스트 케이스와 관련 문서의 품질이 향상

\* **팀의 결속 및 학습 향상**: 팀원 간의 의견 교환을 통해 유대감을 강화하고, 서로의 경험을

배우는 기회를 제공

**\* 테스트 베이스 품질 개선:** 테스트 기준과 요구 사항을 지속적으로 개선함으로써 품질을 높임.

**\* 개발과 테스팅 간의 협업** **개선:** 개발자와 테스터 간의 소통을 촉진하여 협업을 강화

**6.2 회고의 단점**

**\* 시간 소모:** 회고는 일정시간을 필요로 하며, 이는 다른 작업을 지연

**\* 비효율적인 회의:** 회의가 비효율적 진행일 경우 생산적인 피드백을 받기 어려움

**\* 부정적인 감정 유발:** 자신의 실수를 팀이 지적, 비판을 하면 불편하게 느낌

**\* 실행 부족:** 도출된 개선 사항이 실제로 실행되지 않거나 후속조치가 되지 않은 경우 효과가 감소

**\* 편향된 피드백:** 특정 팀원이 주도하는 경우 그들의 의견이나 경험만 반영되며 다양성을 저하

**\* 기업 문화의 저해:** 회고생산을 부정적으로 사용될 경우 팀원 간의 신뢰성을 저하.