소프트웨어공학 활용

애자일 방법론

한국기술교육대학교
온라인평생교육원

학습내용

- 애자일 개요
- 애자일 활용

학습목표

- 애자일 방법론에 대한 개념을 정의할 수 있다.
- 애자일 방법론의 종류를 설명할 수 있다.

애자일 개요

1 애자일 방법론

1 애자일 개념

1. 애자일 개발 프로세스란 어느 특정 개발 방법론을 가리키는 말은 아님

2. Agile(기민한, 좋은 것을 빠르고 낭비 없게 만드는 것) 개발을 가능하게 해주는 다양한 방법론 전체를 일컫는 말

3. 개발과정에서의 시스템의 변경사항을 유연하게 또는 기민하게 대응할 수 있도록 방법론을 제공한다는 것을 의미

4. 큰 하나를 단계별로 작은 여러 개로 나누고, 작은 프로젝트를 하나씩 수행하고 테스트하며 개발과정에서의 시스템 변경사항을 유연하게 또는 기민하게 대응할 수 있도록 고안된 개발 방법론

5. 문서 중심의 산출물보다는 동작하는 소프트웨어 중심

애자일 개요

1 애자일 방법론

2 애자일 선언문 분석

프로세스와 도구보다 개개인과 상호 소통이 더 중요
(Individuals and Interactions over Processes and Tools)

개발자들은 서로 간의 정보를 공유하기 위해 얼굴을 보고 대화해야 함

포괄적인 문서화보다 제대로 동작하는 소프트웨어가 더 중요
(Working Software over Comprehensive Documentation)

문서를 작성할 시간에 완성도 높은 소프트웨어를 개발하고
고객에게 짧은 기간에 주기적으로 제공해야 함

계약 협상보다 고객과의 협력이 더 중요
(Customer Collaboration over Contract Negotiation)

개발자들은 관련된 영역의 사람들과 함께 일해야 함

세워진 계획을 따르기보다는 변화에 대한 대응이 더 중요
(Responding to Change over Following a Plan)

개발 후반부라고 하더라도 고객의 요구사항을 적극적으로 검토 및
반영해야 함

애자일 개요

1 애자일 방법론

3 핵심가치

고객참여	계약 협상에 앞서 고객과의 협력
프로세스보다 사람	프로세스나 도구에 앞서 개인과 상호협력
변경을 포용	계획 준수에 앞서 변화에 대응
소프트웨어 중심	포괄적인 문서화에 앞서 작동하는 소프트웨어
반복적인 릴리스	고객과의 소통을 위해 주기적으로 소프트웨어 제공

애자일 개요

1 애자일 방법론

4 애자일 특징

1. 애자일 소프트웨어 개발은 반복 점진적(Iterative and Incremental) 개발을 기본 스타일로 함
2. 자기조직화(Self-Organizing)나 교차 기능팀(Cross-Functional Teams) 등의 기법들을 활용
3. 미리 모든 것을 알 수 없는 상황이라는 전제 하에 프로젝트 진행
4. 변화에 민첩하게 고객의 요구를 반영, 프로토 타입 제작 및 피드백을 통한 의견 반영, 수정을 반복적으로 수행
5. 짧은 개발 주기를 반복하여 프로젝트가 제대로 진행되는지 확인하고, 이를 통한 위험요소를 최소화함
6. 다른 SDLC 모델들과 다르게 분석과 설계를 크게 강조하지 않음
7. 작업 중인 소프트웨어가 상세 문서보다 더욱 중요시되기 때문에 구현단계를 초반에 수행함

애자일 개요

1 애자일 방법론

5 주요 애자일 방법론 종류

SCRUM | XP(eXtrem Programing) | Lean Software Development | Feature-Driven Development - FDD | 이외의 종류

- 정의자 : 켄 슈와버, 제프 서덜랜드
- 백로그와 스프린트
- 30일마다 동작 가능한 제품을 제공하는 스프린트(Sprint) 중심
- 매일 정해진 시간에 정해진 장소에서 짧은 시간의 개발을 하는 팀을 위한 프로젝트 관리 중심방법론

SCRUM | XP(eXtrem Programing) | Lean Software Development | Feature-Driven Development - FDD | 이외의 종류

- 정의자 : 켄트 벡/에릭 감마
- 의사소통, 단순성, 피드백, 자신감
- 애자일 개발 프로세스의 대표자로 애자일 개발 프로세스 보급에 큰 역할을 함
- 고객과 함께 2주 정도의 반복 개발
- 테스트와 우선 개발을 특징으로 하는 명시적인 기술과 방법을 가짐

애자일 개요

1 애자일 방법론

5 주요 애자일 방법론 종류

SCRUM | XP(eXtrem Programing) | **Lean Software Development** | Feature-Driven Development - FDD | 이외의 종류

- 정의자 : 메릭 포펜딕, 폼 포펜딕
- 도요*(자동차 제조사)의 프로세스를 소프트웨어 개발에 적용한 방법론
- 구체적인 개발 프로세스를 정의하지 않고 철학적인 접근 방식을 정의
- 낭비는 곧 결함이기 때문에 결함을 줄이는 것이 좋은 방법이라는 사고방식

SCRUM | XP(eXtrem Programing) | Lean Software Development | **Feature-Driven Development - FDD** | 이외의 종류

- 정의자 : 피터 코드, 제프 드루카
- 기능 주도 개발 방법
- Feature마다 2주 정도의 반복 개발을 실시
- UML을 이용한 설계 기법과도 밀접하게 관련

애자일 개요

1 애자일 방법론

5 주요 애자일 방법론 종류

SCRUM | XP(eXtrem Programing) | Lean Software Development | Feature-Driven Development - FDD | 이외의 종류

- Adaptive Software Development(ASD) : 적응형 소프트웨어 개발 방법론
- Dynamic Systems Development Method(DSDM) : 동적 시스템 개발 방법론
- Crystal Family
- Agile Unified Process(AUP) : 애자일 UP

애자일 개요

2 전통적인 개발 방법론과 비교 및 필요성

1 기존 방법론과 비교

1 계획수립

전통적 방법

- 상세한 계획 수립, 계획 기반 프로세스
- 다음 단계에 이르기까지의 상세한 계획 수립

애자일 방법론

- 잦은 계획 수립 및 갱신, 경험기반 프로세스
- 바로 다음 반복주기에 대해서만 상세한 계획 수립

2 요구사항 관리

전통적 방법

- 초기 요구사항 수집 및 엄격한 변경관리
- 요구사항 정의 단계에서 모든 요구사항을 정하는 것을 강조

애자일 방법론

- 지속적인 요구사항 개발 및 변경 수용
- 요구사항에 대한 베이스라인 설정을 강조하지 않음

애자일 개요

2 전통적인 개발 방법론과 비교 및 필요성

1 기존 방법론과 비교

3 설계와 테스트 방법

전통적 방법

- 설계 : 상세한 사전(Up-Front) 설계
- 테스트 방법 : 특정 기능이 구현되고 나서야 단위-통합-시스템으로 확장해 나가는 방식

애자일 방법론

- 설계 : 적시(Just In Time) 설계
- 테스트 방법 : 잦은 개발~테스트 주기를 통하여 많은 시간과 비용이 들어가기 전에 기능을 검증

4 문서화와 역할

전통적 방법

- 문서화 : 중량 프로세스 및 상세한 문서화 강조
- 역할 : 엄격한 역할 분리

애자일 방법론

- 문서화 : 경량 프로세스나 문서화보다 코드를 강조
- 역할 : 전체 팀워크를 중요시함

애자일 개요

2 전통적인 개발 방법론과 비교 및 필요성

2 기업들의 애자일 필요성

1. 팀의 생산성을 높이고 제품을 적기에 출시하기 위해
2. 개발에 들어가는 비용을 줄이기 위해
3. 소프트웨어 품질을 향상시키기 위해
4. 팀의 사기와 업무 만족도 향상을 위해

애자일 개요

2 전통적인 개발 방법론과 비교 및 필요성

3 향후 전망

부정적 측면	긍정적 측면
• 방법론 그 자체로서 적용하기에는 프로세스 정립은 부족함 • 대형 프로젝트에 적용하기 적합하지 않음 • 관리 방법에 대한 가이드라인이 부족함	• 방법론 그 자체로서가 아니라, 일부 기법 또는 사상을 선택하여 쓰기에 매우 좋음 • 중·소형 프로젝트에서 적용하기에 적합함 • 대형 프로젝트라 할지라도 특정 Task에 대해 프로세스를 채택하는 것이 바람직한 영역이 있음 • 아키텍쳐 설계 및 프로토타이핑 수립과 같은 태스크 수행 시 적합함

애자일 활용

1 SCRUM

1 스크럼(SCRUM)의 정의

1 개념

스크럼(SCRUM) 이란?

프로젝트 관리를 위한 애자일 방법론

추정 및 조정 기반의 경험적 관리기법의 대표적인 형태

1995년 켄 슈와버와 제프 서덜랜드가 소개로 알려짐

2 역할

제품 책임자
(Product Owner)

- 제품 기능목록에 해당하는 제품 백로그(Product Backlog)를 만듦
- 우선순위를 조정하거나 새로운 항목을 추가하는 일을 관리
- 스프린트에 대한 계획을 수립할 때까지 중요한 역할을 함
- 스프린트가 시작되면 최대한 팀 운영에 관여하지 않는 것을 권장

애자일 활용

1 SCRUM

1 스크럼(SCRUM)의 정의

2 역할

스크럼 마스터
(Scrum Master)

- 문제 해결, 방해요소 제거를 위해 노력
- 스크럼의 원칙과 가치를 지키면서 개발 진행이 가능하도록 지원

스크럼 팀
(Scrum Team)

- 5~9명으로 구성
- 하나의 스프린트 동안 구현해야 할 기능을 구현
- 문제를 보고하며 완료하기 위해 노력

애자일 활용

1 SCRUM

2 스크럼 프로세스

1 스프린트(Sprint)

스프린트(Sprint)란?

달력 기준 1~4주 단위의 반복 개발 기간

2 3가지 미팅

일일 스크럼

- 매일 진행하는 15분간의 프로젝트 진행 상황을 공유하는 회의
- 모든 팀원이 참석하며 매일매일 각자가 한 일, 할 일, 문제점 등을 공유

스프린트 계획

- 각 스프린트에 대한 목표를 세우고 제품 백로그로부터 스프린트에서 진행할 항목을 선택
- 항목별 담당자 배정, 태스크 단위로 계획 수립

스프린트 리뷰

- 스프린트 목표를 달성했는지, 작업 진행과 결과물을 확인하는 회의
- 스프린트 동안 진행된 모든 작업에 대한 데모를 진행

애자일 활용

1 SCRUM

2 스크럼 프로세스

3 3가지 산출물

제품 백로그

- 제품에 담고자 하는 기능의 우선순위를 정리한 목록
- 고객을 대표하여 제품 책임자가 주로 우선순위를 결정

스프린트 백로그

- 하나의 스프린트 동안 개발할 목록으로 사용자 스토리와 이를 완료하기 위한 작업을 테스크로 정의
- 각각의 테스크의 크기는 시간 단위로 추정

소멸 차트

- 개발을 완료하기까지 남은 작업량을 보여주는 그래프
- 각 이터레이션별로 남아있는 작업량을 스토리 포인트라는 것으로 나타냄

애자일 활용

1 SCRUM

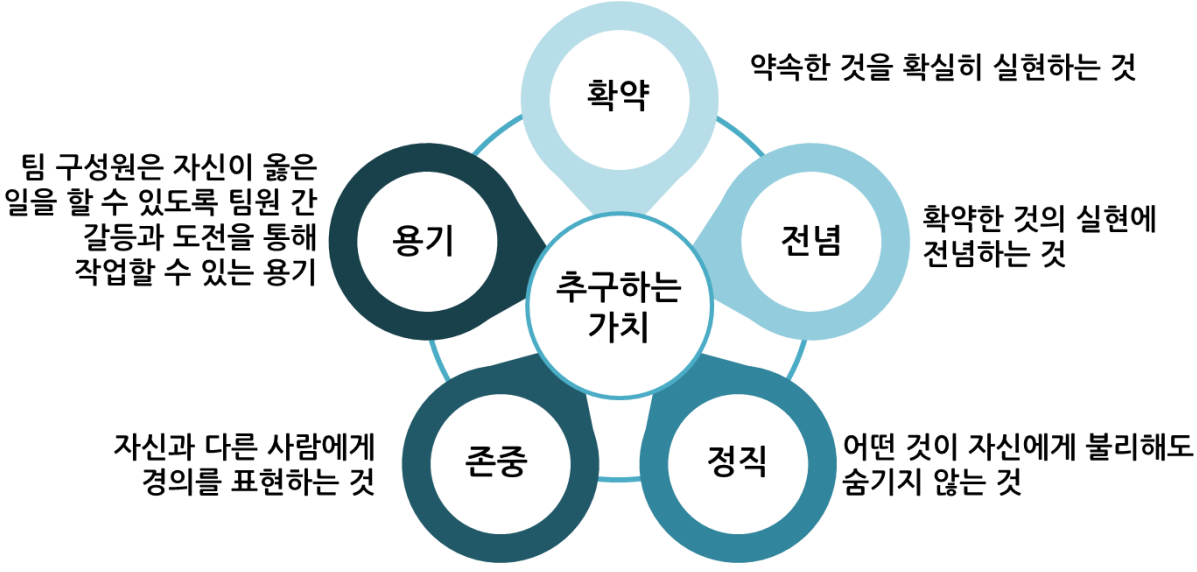
3 스크럼 특징

투명성 (Transparency)	• 스크럼 회의, 소멸차트, 스프린트 리뷰와 같은 기법을 이용하여 다른 방법론에 비해 프로젝트의 상태나 문제점을 파악하기 좋음
타임 박싱 (Time Boxing)	• 스크럼은 매일 15분 짧은 시간만 진행 • 스크럼 자체 진행에 있어서 엄격하게 시간을 제한 프로젝트에만 집중할 수 있도록 함
커뮤니케이션 (Communication)	• 각 개발자 간의 장애, 경험 부족 등의 문제 공유 • 각 개발자 간의 구현 난이도 / 시간을 관리
경험주의 모델 (Inspect & Adapt Model)	• 프로젝트 진행 시 개개인의 경험을 기반 • 프로젝트마다 특정 상황 혹은 특징을 가지고 있으므로 기존의 정형화된 프로세스는 적용이 어려움 • 기본적인 구조는 같으나 프로젝트가 시작되면 팀마다 프로세스의 변경을 허용

애자일 활용

1 SCRUM

4 추구하는 가치



애자일 활용

1 SCRUM

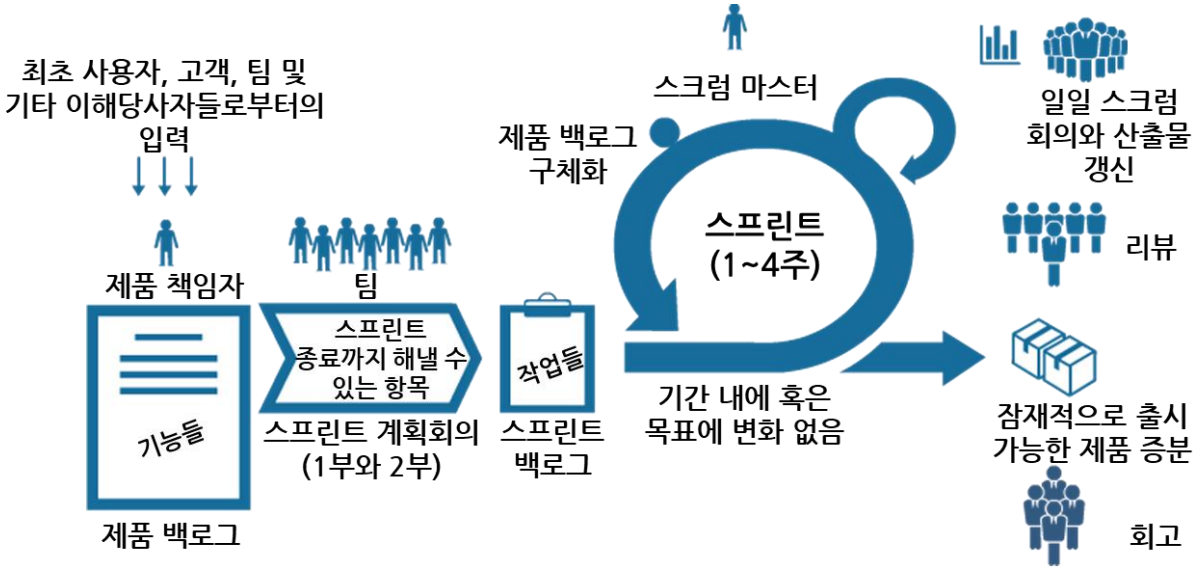
5 스크럼 진행순서

1. 제품에서 요구하는 기능과 우선순위를 제품 백로그로 정함
2. PO(Product Owner)가 정한 제품의 우선순위에서 어디까지 작업을 할지 팀과 조율, 선정된 제품 백로그가 스프린트의 목표가 됨
3. 스프린트 목표를 구현할 수 있도록 팀에서 스프린트 백로그를 작성한 뒤 작업 할당
4. 매일 정해진 장소와 시간에 모든 개발팀원이 참여하는 일일 스크럼 회의를 진행
5. 스프린트가 종료할 때마다, 스프린트 리뷰 미팅을 통해 만들어진 제품을 학습하고 이해
6. 스프린트 회고를 통해 팀의 개발 프로세스에 대한 개선의 시간을 가짐
7. 스프린트 기간 중 다음 스프린트를 준비하기 위해 PO와 필요 인원이 모여 백로그를 준비

애자일 활용

1 SCRUM

5 스크럼 진행순서



애자일 활용

2 XP(eXtreme Programming)

1 XP(eXtreme Programming)란?

1 개념

1. 1990년대 후반 켄트 벡/에릭 감마에 의해 만들어진 개발 기법 중심의 애자일 방법론 중 하나
2. 개발 조직이 중심이 되는 중소규모 팀에 적합한 경량화된 개발 방식
3. 기본적으로 가치(Value)와 그 가치를 이루기 위한 실천(Practice), 이 두 개를 큰 축으로 놓고 원칙(Principle)을 세워 그 둘 사이에서 균형을 맞춤
4. 만들어진 이후 핵심 가치, Practice 등이 진화함

2 목적

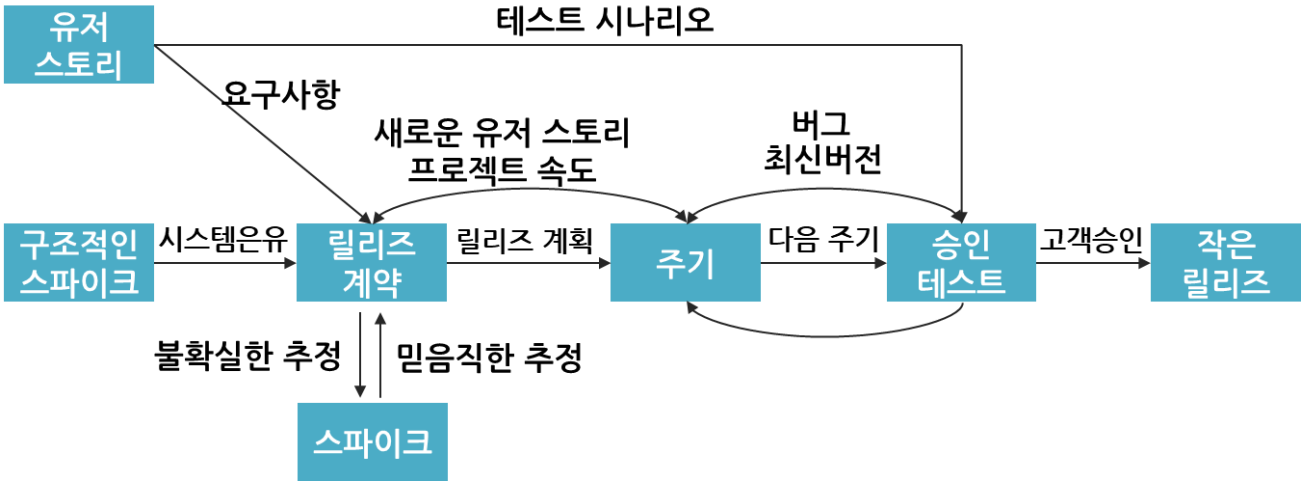
- **Coding**
  - 시스템 개발 활동의 **진짜 중요한 산출물**은 코드임
- **Testing**
  - **단위테스트-코딩**의 불확실성 검증
  - **승인 테스트-의도**에 대한 불확실성 검증
- **Designing**
  - **시스템의 복잡성, 종속성 회피**를 위한 방법
- **Listening**
  - 비즈니스에 의하여 기능 결정
  - **기능 결정을 위하여 비즈니스를 알아야 함**

애자일 활용

2 XP(eXtreme Programming)

2 XP 프로세스

1 프로세스 흐름



2 상세설명

유저스토리

- 고객이 자신이 원하는 내용을 간략하게 기술
- 릴리즈 계획 및 승인 테스트 계획의 토대가 됨

스파이크 솔루션

- 기술적 또는 설계상의 어려운 문제를 해결하기 위한 것
- 기술적인 문제를 줄여 유저스토리에서 추정한 개발 일정의 신뢰도를 높이는 것

릴리즈 계획

- 주기(1~3주)의 시작마다 회의를 통해 처리할 유저 스토리를 고객과 선정
- 고객 관점의 유저스토리를 개발자 관점의 과제로 분리하여 진행

애자일 활용

2 XP(eXtreme Programming)

2 XP 프로세스

2 상세설명

주기

- 릴리즈 계획에 따른 개발의 의미
- 해당 주기가 시작될 때 주기 계획회의를 통해 주기에 해야 할 일 결정

승인 테스트

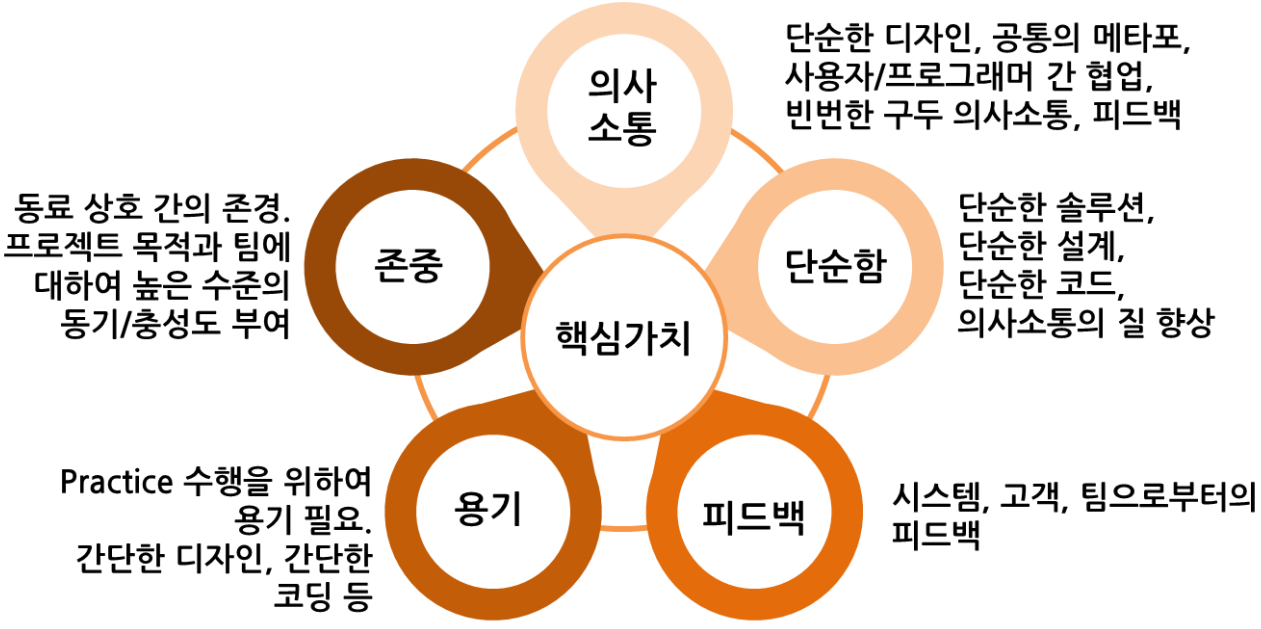
- 각 주기의 유저스토리에 대한 처리 완료 기준

애자일 활용

2 XP(eXtreme Programming)

2 XP 프로세스

3 XP 핵심 가치



애자일 활용

2 XP(eXtreme Programming)

2 XP 프로세스

4 XP Practice

정교한 피드백	페어 프로그래밍, 게임 계획, TDD, 전체 팀
지속적 프로세스	지속적 통합, 리팩토링 또는 디자인 개선, 작은 배포 주기
이해의 공유	코딩 표준, 코드 공동 소유, 간단한 설계, 시스템 메타포
프로그래머 복지	지속할 수 있는 페이스

TDD(Test-Driven Development) : 테스트 주도 개발

두 명의 프로그래머가 함께 프로그래밍 작업을 수행

5 XP 원칙

1. 인간성	소프트웨어는 인간이 개발함
2. 경제성	모든 것에 누군가 돈을 지불한다는 경제성의 원칙을 인정함
3. 상호이익	모든 활동은 관련된 모든 사람에게 이익이 되어야 함

애자일 활용

2 XP(eXtreme Programming)

2 XP 프로세스

5 XP 원칙

4. 자기 유사성	어떤 해결책의 구조를 다른 맥락에도 적용함
5. 개선	소프트웨어 개발에서 '완벽하다'란 말 대신 '완벽해지기 위해 노력한다'만 있음
6. 다양성	팀에는 비록 갈등의 요소가 될 수 있을지라도 다양성이 필요함
7. 반성	좋은 팀은 실수를 숨기지 않고 오히려 실수를 드러내어 거기에서 배울 수 있음
8. 흐름	개발의 모든 단계를 동시에 진행함으로써 가치 있는 SW를 끊임없이 제공함
9. 기회	가끔은 생각을 전환해서 문제를 기회로 보는 방법을 습득함
10. 잉여	소프트웨어 개발에서 핵심적이고 해결하기 어려운 문제는 해결방법을 여러 개 만들어 놓아야 함
11. 실패	성공하는데, 어려움을 겪는다면 실패해도 됨

애자일 활용

2 XP(eXtreme Programming)

2 XP 프로세스

5 XP 원칙

12. 품질	품질을 희생하는 것은 프로젝트 관리의 수단으로 삼기에 효과적이지 않음
13. 책임 소재	어떤 일을 하겠다고 선언한 사람이 그 일의 책임도 함께 가짐
14. 아기걸음	단계를 작게 쪼갤 때 걸리는 부하가 큰 변화를 시도했다가 실패해서 돌아갈 때 드는 낭비보다 훨씬 작음

애자일 활용

3 LSD(Lean Software Development)

1 Lean Software Development

1 개념과 목표

1. 도요* 자동차의 독특한 생산방식(TSP)의 원칙과 실천법을 정리하는 데서 개념이 시작됨

2. JIT(Just In Time)은 필요한 시점에 필요한 만큼만 생산하는 것을 의미
   - 이를 통해 재고를 최소화하고 비용 절감

3. 칸반(Kanban)은 일종의 작업지시로서 Pull 방식의 생산 시스템을 구축하는 데 중요한 역할을 함

4. 불필요한 것을 버리고 효율성을 기해 경량화하는 것

5. 낭비를 발견하고 제거함

시스템의 낭비를 줄이고 고객에게 더 높은 가치를 제공

애자일 활용

3 LSD(Lean Software Development)

2 개발 원칙

1. 낭비를 제거하라	• 파레토법칙에 따라 개발에 정말 중요한 20%에 집중하고 낭비되는 요소를 제거함
2. 품질을 내재화하라	• TDD를 통해 코드의 실수를 방지함 • 빅뱅 통합을 버리고 지속적인 통합과 중첩된 동기화 기법 사용
3. 지식을 창출하라	• 과학적 방법 사용 • 모든 사람이 따라 하고 잘 알려진 실천법을 표준에 포함 • 누구든지 표준에 도전하고 변경하도록 장려
4. 확정을 늦춰라	• 마지막까지 변화를 수용할 수 있도록 코드 작성 • 의존성을 깨뜨리고 옵션을 유지함
5. 전체를 최적화하라	• 고객 요구에서 S/W 배포까지 전체 가치 흐름에 초점을 맞춤

애자일 활용

3 LSD(Lean Software Development)

2 개발 원칙

6. 사람을 존중하라	• 효과적인 리더십 제공 • 팀은 자부심, 책임감, 신뢰, 칭찬을 통해 번성함
7. 빨리 인도하라	• 신속한 인도, 고품질, 저비용은 공존할 수 있음 • 일의 양을 할 수 있는 만큼으로 제한함

애자일 활용

3 LSD(Lean Software Development)

3 낭비

1 개념 및 예

낭비란?

고객에게 가치를 주지 못하는 것

낭비의 예

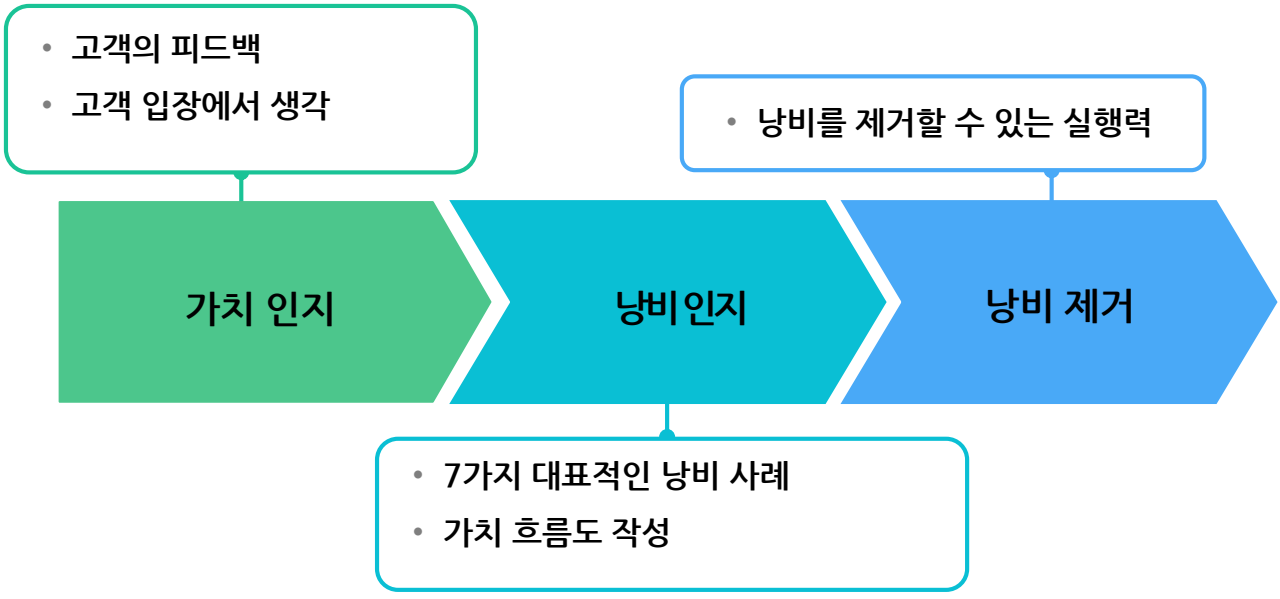
- 미완성 작업 : 배포되지 않은 코드
- 가외기능(Extra Processes) : 필요하지 않은 기능 추가
- 재학습 : 알던 것을 잊었다가 다시 생각해내는 것
- 이관
  - 암묵지가 전수되지 못함
  - 한번 이관 시 지식 및 데이터가 누락
- 작업 전환 : 작업 전환 시간 ⇒ 낭비
- 지연 : 다른 일을 하는 개발자를 기다림
- 결함 : 결함은 부가적인 작업을 추가

애자일 활용

3 LSD(Lean Software Development)

3 낭비

2 낭비 제거 3단계



학습정리

1. 애자일 개요

- 애자일 개발 프로세스란 어느 특정 개발 방법론을 가리키는 말은 아니지만, 개발을 가능하게 해주는 다양한 방법론을 일컫는 말임
- 문서 중심의 산출물보다는 동작하는 소프트웨어 중심인 개발론임
- 애자일의 핵심 가치는 활발한 고객 참여, 프로세스보다는 사람들과의 상호 협력이 중요하고, 고객과의 소통을 주기적으로 하여 고객의 요구사항을 적극적으로 반영한 반복적인 릴리스를 통해 소프트웨어 중심의 가치를 추구함
- 애자일 소프트웨어 개발은 반복 점진적 개발을 기본 스타일로 하여 미리 모든 것을 알 수 없는 상황을 전제 하에 프로젝트를 진행함
- 짧은 개발 주기를 반복하여 프로젝트가 제대로 진행되는지 확인하고 이를 통한 위험 요소를 최소화함
- 기존 개발론과의 차이점은 분석과 설계를 크게 강조하지 않고 실질적인 개발에 중점을 둠
- 애자일 방법론으로는 SCRUM, XP, Lean, FDD 등이 있음

학습정리

2. 애자일 활용

- 스크럼은 프로젝트 관리를 위한 애자일 방법론으로 추정 및 조정 기반의 경험적 관리기법의 대표적인 형태임
- 스크럼의 제품 책임자는 제품 백로그를 만들고 우선순위를 조정하여 새로운 항목들을 추가하는 역할을 함
- 스크럼 마스터는 문제 해결, 방해요소 제거를 위한 노력을 함
- 스크럼 팀은 5~9명으로 구성되며 개발 기간 내에 문제가 발생하면 보고하고 문제 해결 및 개발을 완료하기 위한 노력을 함
- 스크럼의 프로세스는 스프린트, 3가지 미팅, 3가지 산출물로 이루어져 있음
- XP(eXtreme Programmin)는 개발 조직이 중심이 되는 중소규모 팀에 적합한 경량화된 개발 방식임
- 가치(Value)와 이를 이루기 위한 실천(Pratice)을 큰 축으로 보고 이들 사이 원칙(Principle)을 세워 균형을 맞춤
- XP는 의사소통, 단순함, 피드백, 용기, 존중이라는 핵심 가치를 추구함
- Lean Software Development(LSD) 개발론은 도요* 자동차의 독특한 생산방식(TSP) 원칙과 실천법에서 시작되었음
- 불필요한 것(낭비)을 버리고 효율성(가치)을 추구함
- 핵심 목표는 시스템의 낭비를 줄이고 고객에게 더 높은 가치를 제공하는 것임