방법론 가이드

SW중소기업을 위한 경량 개발 방법론

과학기술정보통신부
Ministry of Science and ICT

nipa 정보통신산업진흥원
National IT Industry Promotion Agency

경량 개발 방법론
(리드미, LeDeMe) 가이드

nipa 정보통신산업진흥원 National IT Industry Promotion Agency
STA Global Testing Leader 테스팅컨설팅

발간사

2016년 세계경제포럼(WEF, World Economic Forum)은 전 세계가 당면하게 될 주요 이슈로 4차 산업혁명에 주목했으며, 최근 빅데이터, 클라우드, 인공지능, 사물인터넷 등 융합과 혁신이 빠르게 진행되고 있습니다.

이러한 흐름의 중심에는 소프트웨어가 있습니다. 최근의 소프트웨어는 이전과는 비교할 수 없는 빠른 속도로 변화하고 있습니다. 그러나 전통적 소프트웨어 개발 방법론은 이러한 변화를 빠르게 수용하기 어렵고, 이에 대한 대안으로 경량 개발 방법론이 주목 받고 있습니다.

한 조사에 따르면 해외의 소프트웨어 프로젝트 상당수가 경량 개발 방법론을 활용하고 있으며, 프로젝트 관리 지식 체계를 다루고 있는 PMBOK (Project Management Body of Knowledge)는 2018년 개정판에 경량 개발 방법론을 신규로 포함 시킬 것임을 발표하였습니다.

그러나 기존의 경량 개발 방법론 가이드들은 개념의 이해 위주여서 실제 적용 시 구체적인 실행 방안이 부족하며, 또한 방법론이 추구하는 수평적 조직 문화와 국내 조직 문화와의 이질성 등의 한계로 성공 사례가 적고, 경량 개발 방법론의 대표적인 애자일 및 데브옵스 도입률은 아시아태평양 지역 국가 가운데 가장 낮다는 조사결과도 발표되었습니다.

본 가이드는 기존 경량 개발 방법론의 장점을 모으고 국내 SW 기업의 특성을 반영한 경량 개발 방법론이며, 활용도를 높이기 위해 상세한 설명과, 다양한 사례 및 팁을 수록하였습니다.

아무쪼록 SW 기업들이 본 가이드를 활용하여 비즈니스의 변화에 민첩히 대응하고, 나아가 4차 산업혁명이라는 변화의 중심에 설 수 있기를 바랍니다. 끝으로 본 가이드를 개발하기까지 수고해 주신 관계자 분들께 감사드립니다.

정보통신산업진흥원

원장 윤 종 록

Contents

Contents

CHAPTER 01

가이드 개요

CHAPTER 01

가이드 개요

1.1 서문

오늘의 비즈니스 환경은 인공지능(AI), 빅데이타 등과 같은 디지털 기술의 비약적인 발전으로 인하여 산업간의 경계가 무너지면서 비즈니스의 불확실성이 한층 높아지고 있다. 애자일은 이렇게 변화된 비즈니스 환경에 효과적으로 대응하기 위하여 발생된 소프트웨어 개발 방법론으로 많은 글로벌 기업들이 자신들의 경쟁력 유지를 위하여 애자일을 적극 도입하여 확산하고 있다. 구성원들의 활발한 소통과 협력, 요구사항 변화에 대한 유연성 등 애자일이 추구하는 가치는 소프트웨어는 물론 비즈니스 경쟁력을 높이는 요인으로 작용하고 있기 때문이다. 애자일이 모든 소프트웨어 개발에 효과적으로 적용될 수 있는 것은 아니지만 기존의 개발방식과 함께 상호보완적으로 활용된다면 비즈니스 성과를 높일 수 있다. 반면에 국내 기업들은 일부 스타트업을 제외하고는 여러가지 요인들로 인하여 애자일을 효과적으로 활용하고 있지 못하는 상황으로 나타났다. 이러한 장애요인들에는 국내 현실을 반영한 적용 가이드 부족이나 적절한 기술 및 도구 통합의 어려움 등이 도출되었다.('경량개발방법론 국내외 현황' 참조)

본 가이드는 이러한 장애요인들을 해소하여 국내 중소 기업들이 빠른 비즈니스 변화에 대응할 수 있는 소프트웨어 개발 기반을 조성하는데 도움을 주고자 만들어졌다. 기존에 존재하는 애자일 관련 가이드나 책들이 다소 추상적이고 해외 사례들이 대부분이다보니 우리나라 현실에 맞는 애자일 적용 방법을 찾기가 쉽지 않은 것이 현실이다. 따라서 국내 중소 기업들이 애자일을 도입할 때 나타날 수 있는 시행착오를 최대한 줄일 수 있도록 국내 현실을 반영한 프로세스와 적절한 도구선정에 대한 가이드를 포함하였다. 다음은 장애요인에 대한 해결책을 본 가이드에 반영한 내용들이다.

■ **국내 현실을 반영한 적용 가이드 부족**

- 애자일에서 가장 많이 사용되는 스크럼에서는 제품책임자와 스크럼마스터, 팀원들의 역할만 있고 팀 리더 역할이 없다. 제품개발에 대한 책임 역시 팀원들에게 있다. 이러한 전제조건들은 팀 리더가 중심이 되어 팀을 책임지고 이끄는 국내 중소기업 현실과는 많은 차이가 있어 애자일 적용시 혼란과 어려움이 따랐다.
- 본 가이드에서는 국내 중소기업의 현실을 반영하여 별도의 스크럼마스터를 두지 않고 팀의 권한과 책임을 가지고 있는 팀 리더가 중심이 되어 프랙티스를 주도적으로 수행하는 것으로 기술하였다.

■ **적절한 기술 및 도구 통합의 어려움**

- 애자일 방법론에는 많은 프랙티스와 도구들이 존재하다보니 처음 애자일을 적용하는 조직들은 어떤 기술과 어떤 도구를 사용하는 것이 좋은지 어려움을 느끼는 것이 일반적이다.
- 본 가이드에서는 성숙도를 감안하여 처음 1단계에서 적용해볼 수 있는 프랙티스 및 도구들과 다음 2단계에서 적용할 수 있는 프랙티스 및 도구들을 구분하여 기술하였다. 가이드 내용중에 '시도해볼 것이나 참조

사항'은 다음 2단계에서 수행할 것을 기술한 것이다. 도구는 애자일 조직에서 가장 많이 사용하는 오픈 소스 도구를 선정하여 도구 가이드를 만들었으며 이를 개별 프랙티스와 연계하였다.

끝으로 본 가이드에 기술되어 있는 프랙티스들은 현재 애자일 조직에서 많이 적용되고 있는 활동중의 하나일뿐 모든 팀에게 맞는 것은 아니다. 애자일 관점에서 베스트 프랙티스란 없으며 모든 조직들은 자신들의 상황에 맞는 적절한 방법을 찾아야 한다. 따라서 같은 조직이라도 팀마다 프랙티스들은 다르게 나타날 수 있으며 각 팀들은 지속적인 회고를 통하여 프로세스를 개선시키는 노력이 필요하다. 본 가이드는 애자일 조직으로 가는 첫 출발점이 될 수 있을 것이다.

1.2 가이드 목적과 적용 범위

경량 개발 방법론(리드미, LeDeMe) 가이드의 목적은 중소 기업이 고객을 위해 소프트웨어 개발을 신속하고 효율적이며 효과적으로 개발을 할 수 있도록 도움을 주는 데에 있다.

1.2.1 목적

본 가이드의 목적은 다음과 같다.

- 중소 기업의 소프트웨어 개발팀이 고객 및 사용자에게 동작하는 소프트웨어를 신속하고 효율적으로 제공할 수 있게 한다.
- 개발팀과 고객 및 사용자가 원활한 의사 소통과 함께 꾸준하고 지속 가능한 방법으로 사용자가 필요한 것을 제공하면서 협업할 수 있게 한다.
- 개발팀과 고객이 요구사항이 변경될 수 있다는 것을 인정하면서 프로젝트 도중에 변경 사항을 관리할 수 있게 한다.

1.2.2 적용 범위

본 가이드는 패키지 소프트웨어 및 웹서비스를 제공하는 소프트웨어 개발 조직에 맞게 개발되었으며 외부 고객을 대상으로 하는 소프트웨어 개발인 경우는 고려하지 않았다.

- 패키지 소프트웨어 및 웹서비스 신규 개발
- 기존 패키지 소프트웨어 및 웹서비스 유지보수
- 신규 개발 및 유지보수를 병행하는 경우

1.3 가이드 활용 방법

가이드 앞에는 목차가 있다.

- 주요 목차 – 주요 장의 제목
- 세부 목차 – 절의 제목

가이드 뒤에는 다음 내용이 있다.

- 용어 해설
- 힌트와 팁의 목록 및 참조 페이지
- 사용된 예제 목록 및 참조 페이지

[예제, 힌트 및 팁]

이 가이드 전반에 걸쳐서 예제, 힌트 및 팁이 있다.

- 예제의 형식

예제 . 예제는 어떤 모양인가?

예제는 이런 모양이다.

- 힌트와 팁 형식

Tip 1. 힌트와 팁은 어떤 모양인가?

힌트와 팁은 이런 모양이다.

- 관련된 추가 정보로 하이퍼링크가 있다.
- 가이드 뒤에는 예제 목록 과 힌트와 팁 목록이 있다.

1.4 가이드 요약

경량 개발 방법론은 반복적 점증적 개발 방법론(iterative and incremental development method)이다.

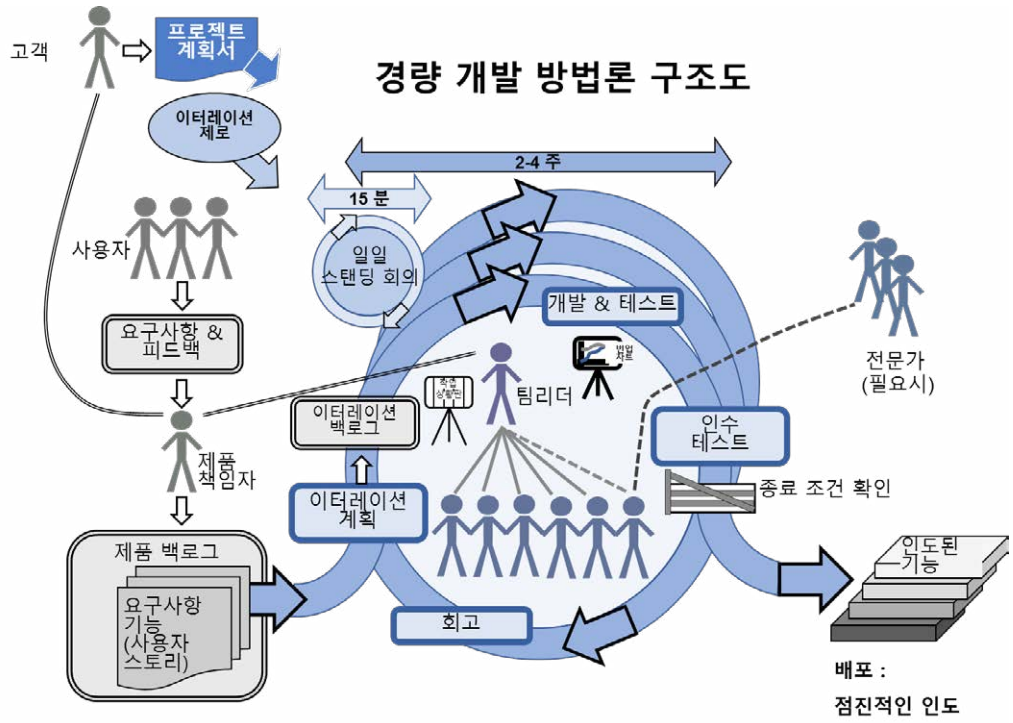
- 반복적 개발 방법론 (iterative development method): 소프트웨어 개발을 진행할 때 제품 개발의 모든 공정(분석, 설계, 구현, 테스트 등)을 여러번 반복하면서 개선해 나간다는 의미이다.
- 점진적 개발 방법론 (incremental development method): 소프트웨어를 인도할 때마다 사용자를 위해 보다 많은 기능에 가치가 부가됨을 의미한다.

Tip 2. 왜 소프트웨어를 인도(deliver)하기 위해 반복적 점진적 방법론을 사용하나?

프로젝트 시작 시에는 고객이 원하거나 사용자가 필요로 하는 것을 완전히 알지 못한다.
- 사용자도 완전히 알지 못하거나 상황이 변할 수도 있다.
- 사용자가 원하는 것을 개발팀이 완전하게 파악하지 못할 수도 있다.
- 사용자가 원하는 것이 그들이 필요로 하는 것이 아닐 수도 있다.

프로젝트 시작 시에 고객이 원하고 사용자가 필요로 하는 모든 것을 파악해 프로젝트가 끝날 때 그것들을 한꺼번에 인도하려고 하면, 심각한 문제가 발생할 수도 있다.

다음 페이지에서 구조도 및 전체 방법론에 대한 설명을 볼 수 있다.



〈 그림 1. 경량 개발 방법론 구조도 〉

시기	활동	해당되는 절 (Section)
프로젝트 시작	**프로젝트 계획**: 프로젝트 계획서를 작성한다. • 프로젝트의 목표 및 그 프로젝트가 제공하는 가치 • 목표 및 인도 횟수와 프로젝트 예산 같은 제약 사항 • 프로젝트의 성공을 측정하는 방법	**프로젝트 계획 수립**
이터레이션 제로	**이터레이션 제로**: 고객과 IT 공급자는 방법론을 사용할 것에 동의하며 또한 • 프로젝트 계획서에 동의한다. • 팀과 협력해 작업할 제품 책임자(고객/사용자 대표)를 정한다. • 무엇을 언제 릴리즈 할 지에 대해 계획한다. • 아키텍처 설계를 한다. • 팀 리더와 팀원을 정한다. • 팀이 일을 시작하는 데 필요한 모든 것을 갖추고 있는지 확인한다.	**이터레이션 제로**
전체 개발기간	**제품 백로그 관리**: 제품 책임자가 사용자의 요구 사항을 확실히 알아야 한다. • 우선 순위를 정해 다음에 어떤 요구 사항이 필요한 지를 결정한다. • 이터레이션 계획에 필요한 요구 사항에 대해 사용자 스토리를 작성한다.	**제품 백로그 관리**
전체 개발기간	**릴리즈 계획**: 제품 책임자와 팀 리더는 사용자 및 고객과 함께 • 인도돼야 할 것을 결정한다. • 제품 릴리즈 시기를 결정한다.	**릴리즈 계획**
이터레이션 시작 시	**이터레이션 계획**: 제품 책임자와 팀(팀 리더 포함)이 만나서 사용자 스토리 중 어떤 것을 해당 이터레이션에 인도할 지 합의한다.	**이터레이션 계획**
이터레이션 동안	**개발 및 테스트**: 팀은 사용자 스토리를 구현하는 소프트웨어를 인도하기 위해 협력해 작업하고 또한 • 일일 스탠드업 미팅을 한다. (설계, 코드 및 단위 테스트를 포함한) 개발을 수행한다. • 해당 소프트웨어를 스토리 테스트한다.	**일일 스탠드업 미팅** **개발** **스토리 테스팅**
매 이터레이션의 종료 시	**이터레이션 마감**: 제품 책임자 및 사용자와 함께 팀은 인도를 위해 작업하며 또한 • 인수 테스팅을 수행한다. • 동작하는 소프트웨어를 배포한다. • 지속적인 개선 제안을 위해 회고 미팅을 한다.	**인수 테스팅** **배포** **회고 미팅**
배포 후	**운영 및 유지보수**: 운영팀과 사용자가 함께 작업하며 또한 • 해당 소프트웨어를 사용해 피드백을 제공한다. • 필요에 따라 소프트웨어를 업데이트한다.	**운영 및 유지보수**

〈 표 1 방법론 요약 〉

CHAPTER 02

방법론 원리 및 권장사항

CHAPTER 02 방법론 원리 및 권장사항

2.1 리드미 원리

리드미 원리는 애자일 소프트웨어 개발 선언문(http://agilemanifesto.org)을 참조하였으며 국내 상황을 고려하여 아래와 같이 정의하였다.

1. 우리의 최고 우선 순위는 가치 있는 소프트웨어를 일찍 그리고 지속적으로 전달함으로써 고객을 만족시키는 것이다.
2. 비록 개발 후반부일지라도 요구사항 변경을 환영하라. 애자일 프로세스들은 변화를 활용해 고객의 경쟁력에 도움이 되게 한다.
3. 동작하는 소프트웨어를 2주에서 4주마다 자주 전달한다.
4. 팀 내에서 정보를 전달하는 가장 효율적이고 효과적인 방법은 면대면 대화이다.
5. 사용자, 제품 책임자, 팀은 프로젝트 내내 함께 작업해야 한다
6. 프로젝트는 작고, 동기 부여되고, 협력적인 팀으로 구성하라. 팀에게 환경을 제공하며 필요한 것을 지원하고, 일을 끝낼 수 있도록 팀을 신뢰하라.
7. 인수된 소프트웨어가 진척의 주된 척도이다.
8. 프로세스는 지속 가능한 개발을 촉진한다. 사용자, 제품 책임자, 팀은 일정한 속도를 계속 유지할 수 있어야 한다.
9. 사용자 요구, 기술적 탁월함, 좋은 설계에 대한 지속적 관심이 기민함을 향상시킨다.
10. 정기적으로, 팀 차원에서 어떻게 하면 더 효과적일 수 있을지에 대해 되돌아보며 자신들의 행동을 이에 따르도록 조율하고 조정한다.

원리, 권장사항은 방법론 수행의 출발점이며 조직 및 프로젝트 상황에 따라 테일러링 할 수 있다.

2.2 권장사항

• 권장사항 •

역할을 위한 권장사항

1. 고객은 제품 책임자와 팀에게 프로젝트 목표를 제공한다.
2. 팀원이 제품 요구사항을 확인하기 위해 이해관계자를 만날 수 있지만, 팀의 주 의사소통 채널은 팀 리더이다.
3. 팀은 팀 리더를 포함해 3~10명으로 구성돼야 한다.
4. 제품 책임자는 제품 요구사항을 설명할 수 있어야 한다. 필요 시 대리인이 선임될 수 있다.
5. 제품 책임자는 제품 백로그에 책임이 있다. 특정한 활동은 팀에 위임할 수 있다.
6. 팀 리더는 개발팀이 효과적이고 효율적으로 수행하도록 보장해야 한다.
7. 팀 리더는 작업 현황판을 사용해 이터레이션 내의 진행상황을 이해관계자가 볼 수 있도록 해야 한다.
8. 릴리즈가 정의될 때, 팀 리더는 릴리즈 번-업 차트를 사용해, 다음 릴리즈에 대한 진행상황을 이해관계자가 볼 수 있도록 해야 한다.
9. 팀원은 이터레이션 계획, 일일 스탠드 업 미팅, 회고 미팅에 참여해야 한다.
10. 팀 리더는 사용자 스토리 개발에 기여해야 한다.
11. 제품 책임자는 팀원으로부터의 질문에 신속하게 응답해야 한다.

활동을 위한 권장사항

12. 이터레이션 기간은 4주를 초과하면 안 된다.
13. 15분으로 제한된 일일 스탠드 업 미팅은 매일 열려야 하며 같은 시간에 개최돼야 한다.
14. 이터레이션 길이는 특별한 상황이 없는 한 변하지 않아야 한다.

작업 산출물을 위한 권장사항

15. 제품 백로그는 모든 이해관계자가 볼 수 있어야 한다.
16. 이터레이션 백로그에서의 사용자 스토리는 한 이터레이션 내에 완료돼야 한다. 완료의 의미는 완료 정의를 충족하는 것이다.

첫 프로젝트일 경우 권장사항

17. 팀원에게 다수의 업무를 부여하지 말아야 한다.
18. 방법론을 교육 받을 수 있도록 프로젝트 워크숍을 가져야 한다.
19. 이터레이션 제로 동안 제품 책임자와 사용자를 위한 교육을 포함한다.
20. 제품 책임자와 팀 리더의 역할을 별개로 한다.
21. 팀을 한 장소에 위치시키고 팀에 최소한 각각 한 명의 디자이너, 개발자 그리고 테스터를 포함시킨다.

팀 공간을 위한 권장사항

22. 가능하면 팀원들은 한 장소에서 일한다.
23. 다른 팀의 방해와 소음으로부터 팀을 보호한다.
24. 팀원을 위한 충분한 작업 공간 및 보관 공간을 확보한다.
25. 작업 현황판과 릴리즈 번-업 차트가 명확히 보이기 위한 적합한 장소를 확보한다.
26. 최소한의 도구와 자원으로 시작하고, 다른 것이 필요하면, 그 때 계획을 세운다.

CHAPTER 03

역 할

CHAPTER 03 역 할

3.1 팀

팀

■ 역할

사용자에게 동작하는 소프트웨어 인도하기

■ 설명

사용자 요구 사항을 인도 가능한 코드로 변환하는 데 필요한 모든 기술을 갖춘 소프트웨어 엔지니어팀. 이 팀은 제품 책임자 및 사용자와 긴밀하게 협력해 작업하며 팀 리더와 팀원(2명에서 9명의)들로 구성된다.

■ 활동

- 이터레이션 계획-4.5
- 기술 및 UX설계
- 코딩 및 단위 테스팅-5.7
- 스토리 테스팅 및 회귀 테스팅- 4.8
- 인수 테스팅-4.9
- 일일 스탠드업 미팅 및 회고 미팅-4.6

■ 작업 산출물

- 설계 산출물
- 코드
- 인수 테스트 세트 및 결과-6.10
- 작업상황판-6.8
- 릴리즈 번-업 차트-6.4
- 인수 테스트 스크립트

3.1.1 역할

팀의 역할은 사용자의 요구를 충족시키는 동작하는 소프트웨어를 인도하는 것이다. 이것은 작업 코드 일뿐만 아니라 사용자가 제품을 성공적으로 사용하기 위해 필요한 모든 것을 의미한다.
팀은 기존의 IT 팀과는 다르다. 왜냐하면 모든 IT 분야 출신의 사람들이 사용자와 함께 서로 긴밀히 협력해 일하기 때문이다. 프로젝트에는 독자적으로 일하는 별도의 팀이 없어야 한다. 즉 하나의 공동 작업 팀만 있어야 하며 산출되는 모든 것에 공동 책임을 져야 한다. 이들은 인도 가능한 산출물의 품질을 집단적 및 개별적으로 책임지며 프로세스(예: 개발, 테스트 및 배포)를 변경할 수 있는 권한을 부여 받는다.
팀은 다양한 스킬(skill)로 소프트웨어 설계, 개발, 테스팅, 배포 및 운영에 기여한다. 이들은 또한 제품 책임자, 사용자 및 고객과 협력해 작업하면서 의사 소통하는 일에 능숙해야 한다.
팀은 하나의 프로젝트 대해서 작업하는 것이 지속성을 유지하는 데 좋다. 팀은 팀 작업실(Team room)에 함께 있으면서 협력해 작업하기 때문에 의사 소통에 도움이 된다.

3.1.2 팀원

팀은 팀 리더와 여러 명의 팀원으로 구성되며 프로젝트에 필요한 다양한 기술과 지식을 공유한다. 이상적인 팀 규모는 7명이며 최소 3명에서 최대 10명이다. 팀에서 필요로 할 기술과 지식은 다음과 같다.

- 기술 설계 및 UX(user experience) 설계
- 사업 분석 및 사업 영역에 관한 지식
- 코딩 및 빌드(build)
- 테스팅, 자동화 및 도구
- 인도(delivery) 및 배포(deployment)
- 문제 해결, 의사 소통 및 대인 관계
- 평가, 보고 및 관리 기술

때로는 전문적 기술을 필요로 하기 때문에, 파트 타임 기준으로 기술 전문가(예: 성능 튜닝, 보안 테스터, 아키텍트(architect), 데이터베이스 전문가)가 참여할 수 있다.

팀 역할에 대한 자세한 내용은 여기를 참조하라.

- **팀 리더**
- **팀원**
- **전문가**

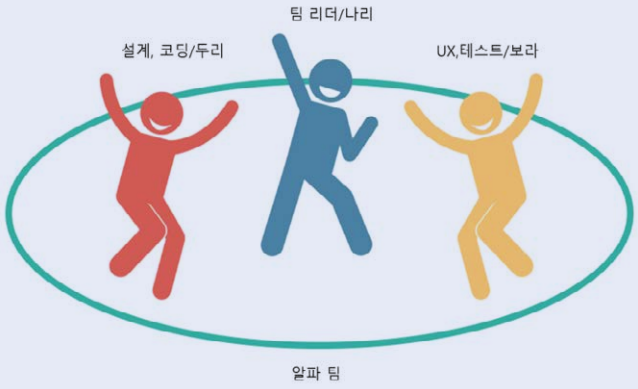
Tip 3. 팀 규모와 개인 스킬(skill) 세트

가장 작은 팀은 3명, 즉 팀 리더와 2명의 팀원이다. 가장 큰 팀은 10명, 즉 팀 리더와 9명의 팀원이다. 팀원마다 각기 팀에 다양한 스킬과 경험을 제공하지만 팀 전체의 스킬 세트는 팀이 수행하는 개발, 테스팅 및 배포(deployment) 활동 전체를 포괄해야 한다.

예제 1. 구성원이 3명인 팀과 10명인 팀의 비교

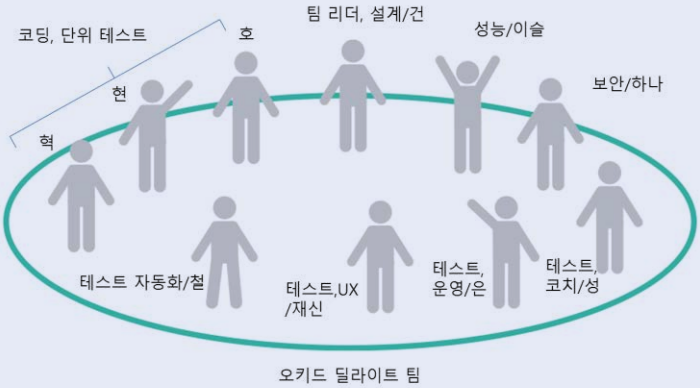
▶ 알파(Alpha)' 팀에는 3명이 있다

'나리'가 팀 리더다. 그녀는 기술 전문가이기도 하기 때문에 기술 설계 결정을 내리고, 다른 2명의 팀원을 코치한다. '보라'는 UX(user experience) 지식을 보유하고 있으며 테스터이다. 그녀는 코딩도 할 수 있다. '두리'는 훌륭한 코더(coder)이며 기술 설계와 테스트를 할 수 있다. 팀원은 자신의 스킬을 향상시켜서 모두가 코드를 작성하고 서로의 작업을 테스트할 수 있어야 한다. 그들은 서로를 돕고 코치하게 될 것이다.



▶ '오키드딜라이츠(OrchidDelights)' 팀에는 10명이 있다.

'건'은 팀 리더이자 기술 전문가이므로 기술 설계를 수행하고 개발자들을 코치한다. '호', '현', '혁'은 코딩 및 단위 테스팅 기술을 보유하고 있으며 개발 작업을 수행하고 있다. '철'은 테스트 자동화를 위한 훌륭한 코딩 기술을 갖춘 기술 테스터이다. '재신', '은', '성'은 테스터이다. '성'은 경험이 가장 많기 때문에 테스팅 기술에 관해서도 모두를 코치할 것이다. '재신'도 UX 지식을 보유하고 있고 '은'은 운영 및 인도 경험을 보유하고 있어서 이 분야들에 집중할 것이다. '이슬'은 성능 튜닝 전문가이며 '하나'는 보안 전문가이다. 이들은 필요할 때만 팀과 함께 한다.



3.2 제품 책임자

제품 책임자

■ 역할

프로젝트에 대한 고객 및 사용자 관점 제시

■ 설명

제품 책임자(PO)는 사용자 스토리로 설명되는 사용자 요구 사항의 주 제시자이자 권한 대행자이다. 제품 책임자는 이터레이션 백로그(Iteration Backlog)의 내용과 릴리즈될 코드를 결정한다.
제품 책임자는 팀, 사용자 및 고객과 긴밀하게 협력한다.

■ 활동

- 제품 백로그 도출 및 관리–4.3
- 이터레이션 제로–4.2
- 릴리즈 계획–4.4
- 이터레이션 계획–4.5
- 인수 테스팅–4.9

■ 작업 산출물

- 제품 백로그–6.2
- 사용자 스토리–6.6
- 이터레이션 백로그–6.5

3.2.1 역할

제품 책임자는 프로젝트에 대한 고객 및 사용자의 관점을 제시할 뿐만 아니라 인도 가능한 소프트웨어의 내용에 대한 결정권도 가지고 있다. 제품 책임자와의 접촉을 통해 팀은 고객 및 사용자의 견해와 "지속적 상호작용(continuous interaction)"을 유지할 수 있다.
제품 책임자는 각 이터레이션이 시작될 때 어떤 사용자 스토리(User Story)를 먼저 개발할 지를 결정하고 이터레이션이 끝날 때 해당 소프트웨어가 배포될 준비가 되었는지 여부를 결정한다. 제품 책임자는 각 사용자 스토리에 대해 팀의 질문에 답변한다. 제품 책임자는 팀, 고객 및 사용자와 긴밀하게 협력한다.

3.2.2 제품 책임자 활동

제품 책임자는 팀이 고객 및 사용자가 필요로 하는 것을 파악하도록 해야 하고 고객 및 사용자에게 진행 상황에 대한 정보를 전달해야 한다. 제품 책임자는 다음을 수행한다.

■ **이터레이션 제로에 참여해 다음 사항들에 대한 책임을 진다.**
- 프로젝트 계획서에 대해 고객 및 팀 리더와 합의한다.
- 초기의 '제품 백로그 관리'를 수행한다.
- 초기의 '릴리즈 계획'을 수행한다.
- 팀 리더와 함께 아키텍처 및 프로세스에 대한 초기의 결정을 확인한다.
- 팀과 고객, 사용자가 서로를 이해하도록 돕는다.

■ **이터레이션 제로 동안 '릴리즈 계획'을 책임지며 그 후 필요에 따라 다음을 수행한다.**
- 고객, 사용자 및 팀 리더와 함께 인도될 기능과 횟수를 결정한다.

■ **이터레이션 제로 동안 '제품 백로그 관리'를 수행하고 그 후 필요에 따라 다음을 수행한다.**
- 고객 및 사용자의 새로운 요구 사항에 따라 사용자 스토리를 제품 백로그에 추가한다.
- 사용하지 않는 사용자 스토리를 제거한다.
- 사용자 스토리가 완전한 지 확인한다.

■ **이터레이션 계획미팅에 참여해 다음을 수행한다.**
- 해당 이터레이션 동안에 다루어지기 원하는 사용자 스토리를 식별한다.
- 팀 리더 및 팀과 협의해 올바른 이터레이션 내용을 합의한다.

■ **인수 테스팅에 참여해 다음을 수행한다.**
- 테스트를 수행하고 사용자가 사용자 스토리에 대한 테스트를 수행하도록 돕는다.
- 테스트된 사용자 스토리에 대한 코드가 릴리즈돼야 할 지 결정한다.

3.2.3 제품 책임자 작업 산출물

제품 책임자의 역할은 고객, 사용자 및 팀이 문서를 작성하고 요구 사항을 파악해 인도하는 데 도움이 되는 실질적인 작업 산출물을 만드는 것이다. 제품 책임자의 작업 산출물은 아래와 같다.

- 제품 백로그
- 고객 및 사용자 요구 사항을 설명하는 사용자 스토리 카드
- 각 이터레이션의 이터레이션 백로그

3.2.4 제품 책임자 기술 및 지식

제품 책임자는 사업 또는 사업 영역에 관한 지식이 있어야 한다. 제품 책임자는 고객과 사용자를 이해해야 한다. 즉 그들의 목표, 업무, 불만 및 효과적이고 효율적인 작업을 위해 그들이 필요로 하는 것들을 파악해야 한다. 제품 책임자는 좋은 의사 소통자가 돼야 하고, 협상에 능해야 하며 다양한 사람들의 요구에 대해 균형을 유지할 수 있어야 한다. 제품 책임자의 결정은 존중돼야 한다.

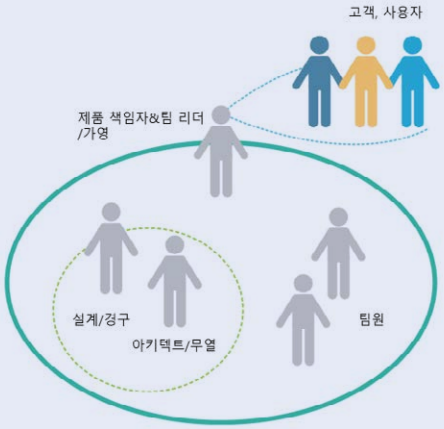
Tip 4. 제품 책임자가 너무 바빠서 지속적 상호작용(continuous interaction)을 하기 어려운 경우에는 어떻게 해야 하나?

질문에 답하고 팀에 정보를 제공하기 위해서는 제품 책임자가 항상 접촉할 수 있는 상태에 있어야 한다. 그러나 제품 책임자는 다른 할 일도 있으므로 항상 접촉할 수는 없다. 제품 책임자가 여러 프로젝트를 담당하고 있거나 다른 할 일이 있다면 제품 책임자와 팀 리더는 서로 협력해 작업할 방법에 대해 협의해야 한다. 며칠마다 미팅을 가질 것인지 외에도 제품 책임자가 격일로 방문하거나 다른 방법의 참여에 동의함으로써 이루어질 수 있다.

예제 2. 제품 책임자 역할을 대체하는 방법 – 팀 리더가 제품 책임자 역할을 함

▶ 팀리더가 제품책임자 역할을 할 수도 있다.

고객과 사용자가 너무 바빠서 '감마(Gamma)' 프로젝트에 대한 제품 책임자 역할을 수행할 수 없어서 팀 리더인 '가영'이 팀 리더 및 제품 책임자 역할 모두를 담당한다. 그녀는 고객과 사용자를 정기적으로 만나서 진행 상황을 보고하고 그들의 요구 사항을 경청한다. 그녀는 그 요구 사항을 사용자 스토리로 변형시키고 사용자를 도와서 인수 기준에 합의한다. 그녀는 팀과 함께 이터레이션 계획을 이끌어 간다. '가영'은 제품 책임자 역할에 중점을 두고 있기 때문에 '경구'를 기술 책임자로 지명해 설계 작업을 수행하게 한다. '경구'는 전문적 아키텍트인 '무열'의 도움을 받는다.



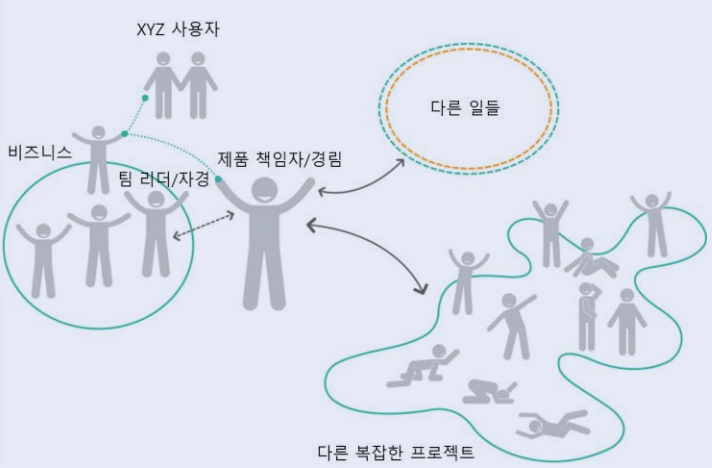
예제 3. 제품 책임자 역할을 대체하는 방법

▶ '자경'은 팀 리더이자 기술적 리더이다.

'자경'은 팀 리더다. 그녀는 기술 전문가이기도 하기 때문에 기술 설계를 수행하고 개발과 테스팅 기술을 담당하는 4명의 팀원을 코치한다.

▶ 제품 책임자가 여러 프로젝트를 담당한다.

'경림'은 'XYZ' 프로젝트의 제품 책임자이지만 사용자로서의 역할도 담당하고 있으며 좀 더 복잡한 프로젝트의 제품 책임자이기도 하다. 그는 이터레이션 계획에 관해 접촉할 것과, 어떤 것을 분명히 할 필요가 있을 경우 'XYZ' 팀과 전화로 접촉할 것을, 팀 리더인 '자경'과 합의한다. 그는 매 이터레이션이 끝날 때마다 1일간의 인수 테스팅을 수행한다. '자경'은 'XYZ' 팀원 중 한 명이 비즈니스 스킬을 보유하고 있는지 확인해 '경림'과 접촉할 수 없을 경우 그를 대신하게 할 것이다. 이렇게 할 경우, '경림'은 보다 복잡한 프로젝트에 집중할 수 있다.



3.3 팀 리더

팀 리더

■ 역할

팀 관리

■ 설명

팀 리더는 팀이 수행하는 작업을 관리하며 팀의 초기 연락 담당자 역할을 수행한다. 팀 리더는 설계에 관한 팀의 결정에 대한 기술적 승인자이다.
팀 리더는 팀 및 제품 책임자와 가장 긴밀하게 협력해 작업한다.

■ 활동

- 이터레이션 제로-4.2
- 릴리즈 계획-4.4
- 이터레이션 계획-4.5
- 일일 스탠드업 미팅-4.6
- 개발-4.7
- 테스팅-4.8
- 인수 테스팅-4.9
- 회고-4.11

■ 작업 산출물

- 작업상황판-6.8
- 릴리즈 번-업 차트-6.4

3.3.1 역할

팀 리더는 팀이 수행하는 작업을 관리하며 팀의 초기 연락 담당자 역할을 수행한다. 팀 리더는 일반적으로 설계에 관한 팀의 결정에 대한 기술적 승인자이다. 팀 리더는 제품 책임자와 긴밀히 협력하며 올바른 작업이 적시에 이루어지는 지를 확인한다. 팀 리더는 팀을 돌보며 팀의 사기, 동기 및 학습에 대해 생각한다.

3.3.2 팀 리더 활동

팀 리더는 팀을 관리하고 릴리즈 계획 같은 상위 수준의 프로젝트 관리 활동을 지원하며 시스템의 기본 설계를 담당한다. 팀 리더는 제품 책임자와 팀원 간의 주 연락자 역할을 수행한다.
팀 리더는 다음과 같은 일을 수행한다.

■ **이터레이션 제로에 참여해 다음 사항들을 담당한다.**
- 각기 다른 기능들을 구현하는 데 걸리는 시간에 대한 아이디어를 제공함으로써 제품 책임자가 최초 릴리즈 계획을 세우는 일을 돕는다.
- 아키텍처 설계(architectural design)가 합의되는 것이 보장되도록 아키텍처 및 설계 전문가와 협업한다.
- 팀에 필요한 인원을 식별해 선발한다.
- 팀에 필요할 모든 자원(팀 작업실, 데스크, IT 환경, 장비 및 도구, 암호, 보안 등)을 식별해 확보한다.

■ **이터레이션 제로 동안 제품 책임자가 릴리즈 계획을 세우는 일을 돕고 그 후 필요에 따라 다음을 수행한다.**
- 제품 책임자, 고객 및 사용자와 함께 인도할 기능 및 횟수를 결정한다.

■ **이터레이션 계획 미팅에 참여해 다음을 수행한다.**
- 사용자 스토리 채택 대상을 검토한다.
- 제품 책임자와 협의하고 팀과 상의해 올바른 이터레이션 내용에 대해 합의한다.

■ **일일 스탠드업 미팅을 수행해 팀원과 의사 소통하고 모든 구성원이 한 사람도 제외되지 않고 필요한 작업에 임하고 있는지를 확인하며 작업상황판(Taskboard)이 최신 상태여서 팀원 및 모든 이해 관계자가 진행 상황을 알 수 있는 지를 체크한다.**

■ **개발, 테스팅 및 배포 작업(특정 기술 세트에 따라 다른 조합)을 수행하고 필요에 따라 팀원을 코치해 자신들의 개발, 테스팅 및 배포 작업을 수행할 수 있도록 한다. 팀 리더는 일반적으로 시스템의 전반적 설계를 담당해 기술적 리더 역할을 수행한다.**

■ **인수 테스팅을 감독한다. 제품 책임자와 사용자가 인수 테스팅에 참여하도록 요구하고 테스트를 수행하는 데 필요한 모든 것이 갖추어져 있는지 확인하며 인수 기준(Acceptance Criteria)과 완료 정의(Definition of Done)를 충족시키기 위해 결함 수정이 수행되도록 한다. 일단 소프트웨어가 인수돼 배포되면 릴리즈 번-업 차트를 업데이트한다.**

■ **회고 미팅(Retrospective Meeting)을 이끌면서 다음을 수행한다.**

- 프로세스에 대한 개선 사항을 식별하기 위해 회고 미팅을 가진다.
- 합의를 통해 개선을 책임지고 이루어냅니다.

3.3.3 팀 리더 작업 산출물

팀 리더는 **작업상황판**과 **릴리즈 번-업 차트**를 담당한다.

3.3.4 팀 리더 기술 및 지식

팀 리더는 팀을 관리하면서 동기를 부여하고 보조하는 일에 능숙해야 한다. 팀 리더는 요구되는 것을 팀이 인도하고 있는지 확인하기 위해 프로젝트 이해 관계자들 특히 제품 책임자와의 의사 소통에 능숙해야 한다. 이상적으로는 팀 리더가 기술 전문가로서 설계, 코딩 및 테스팅 활동을 이끌면서 팀이 필요로 하는 기술적 스킬을 코치한다.

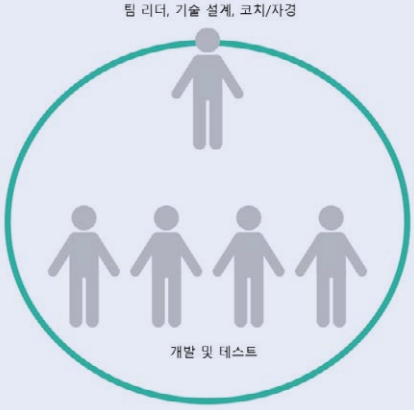
Tip 5. 팀 리더 및 기술적 전문 지식

이상적으로는 팀 리더가 팀에서 가장 경험이 많은 최고의 기술자이다. 이 경우 팀 리더는 기술 설계와 기술 과제에 대해 팀원을 코치하는 일도 담당할 것이다. 그러나 팀 리더가 기술 전문가가 아닌 경우도 있다. 팀 리더가 팀 관리와 제품 책임자와의 협력, 즉 동기 부여, 의사 소통, 협상 등에 능숙하기 때문에 팀을 이끌 수도 있다. 이 경우 팀 리더는 팀 내의 다른 구성원에게 기술 설계 및 기술 코칭을 위임해야 한다.

예제 4. 팀 리더가 기술적 설계의 리더인 경우

▶ '자경'은 팀 리더이자 기술적 리더이다.

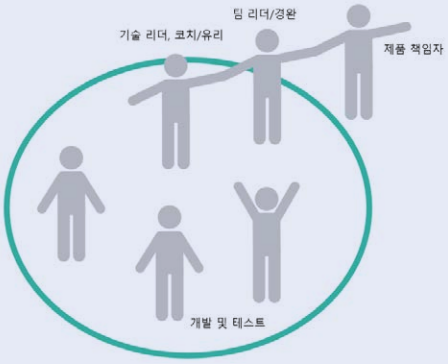
'자경'은 팀 리더다. 그녀는 기술 전문가이기도 하기 때문에 기술 설계를 수행하고 개발과 테스팅 기술을 담당하는 4명의 팀원을 코치한다.



예제 5. 팀 리더가 기술적 설계의 리더가 아닌 경우

▶ '경완'은 팀 리더고 '유리'는 기술 리더이다.

경완은 'A1' 프로젝트의 팀 리더다. 그는 제품 책임자 및 고객, 사용자와 협력하는 일에 능숙하다. 그는 해당 비즈니스를 파악하고 있다. 그는 '유리'를 기술적 설계의 리더이자 코치로 임명한다. 다른 팀원은 개발과 테스팅 기술을 담당하고 있다.



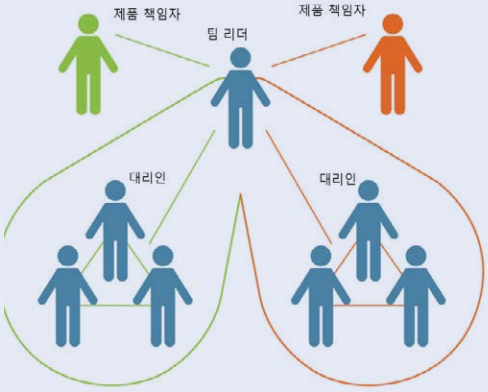
Tip 6. 단일 또는 다중 프로젝트 참여

이상적으로는 팀(팀 리더 및 팀원)이 한 번에 하나의 개발 프로젝트에 대해서만 작업한다(배포된 시스템에 대한 비정기적인 유지보수를 수행해야 할 수도 있다). 이것은 연속성을 제공해 이들이 단 하나의 주제에 집중할 수 있게 하는데, 이는 소프트웨어 엔지니어들이 정기적으로 프로젝트들 간에 '문맥 전환(context switch)'을 해야 하는 경우보다 훨씬 생산적이다. 하지만 때로 팀 리더는 서로 다른 2개의 팀을 이끌면서 하나 이상의 프로젝트에 대해 작업하도록 요구 받는다.

예제 6. 리더가 2개의 팀을 운영하는 경우

▶ 팀 리더가 2명의 제품 책임자와 협력하면서 2개의 팀을 운영하고 있다.

'경완'은 'A1' 프로젝트의 팀 리더이자 'B2' 프로젝트의 팀 리더이기도 하다. 그는 2명의 제품 책임자와 협력하면서 2개의 팀을 운영하고 있다. 그는 이들과의 작업을 위해 자신의 시간을 분할한다. 그는 각 팀의 가장 선임자(한 팀에서는 개발자, 다른 팀에서는 테스터)를 그의 대리자로 지명해 자신이 다른 팀과 작업하고 있을 경우 결정권을 그들에게 부여한다.



3.4 팀 원

팀 원

■ 역할

인도 가능한 코드의 설계, 프로그래밍, 테스팅 및 배포

■ 설명

팀원은 사용자 스토리를, 설계, 프로그래밍, 테스팅 및 배포를 포함해, 인도될 코드로 변환하는 데 필요한 다양한 작업을 수행한다. 팀원은 팀 리더에 의해 관리되며, 다른 팀원, 제품 책임자 및 사용자와 긴밀하게 협력하면서 작업한다.

■ 활동

- 이터레이션 계획-4.5
- 일일 스탠드업 미팅-4.6
- 개발-4.7
- 테스팅-4.8
- 인수 테스팅-4.9
- 배포-4.10
- 회고-4.11

■ 작업 산출물

- 설계 문서
- 인도 가능한 코드
- 테스트 및 그 결과

3.4.1 역할

팀원의 역할은 동작하는 소프트웨어가 사용자에게 인도되도록 돕는 것이다. 팀원은 팀의 목표 달성에 도움이 될 개별 기술들의 조합을 이끌어 낸다. 일반적인 기술은 설계, 프로그래밍, 테스팅 및 배포에 관한 것들이며 각 팀원은 이들 분야의 다양한 기술을 효과적으로 조합한다.
팀은 3~10명의 팀원(팀 리더 및 2~9명의 팀원)으로 구성되며, 제품 책임자와 합의한 대로, 이터레이션 백로그를 위해 선택된 사용자 스토리에 포함되는 모든 기능을 인도할 수 있도록 함께 작업한다.
팀원은 팀원, 제품 책임자 및 사용자 간의 의사 소통을 통해 협력하면서 공동 작업을 수행한다.

3.4.2 팀원 활동

팀원은 동작하는 소프트웨어를 인도할 수 있게 하기 위해 여러 활동을 수행한다. 팀원은 팀 리더 및 다른 팀원에게 자신이 하고 있는 일, 진행 상황 및 진행을 방해하는 사항에 대한 정보를 계속 제공해야 한다.
팀원은 다음의 일들을 수행한다.

■ **이터레이션 계획 미팅에 참여해 다음을 수행한다.**
- 사용자 스토리의 사이즈를 추정해 이것이 다음 이터레이션 동안 완료될 수 있을 지의 여부를 제품 책임자가 결정하도록 돕는다.
- 제품 책임자, 팀 리더 및 팀원과 협력해 올바른 이터레이션 내용에 합의한다.

■ **일일 스탠드업 미팅에 참여해 다음을 수행한다.**
- 자신의 작업 진행 상황과 완료된 작업을 보고한다.
- 도움이 필요한 진행 방해 사항에 대해 보고한다.
- 자신의 다음 작업에 대해 보고한다.
- 팀 전체의 진행 상황을 파악하기 위해 다른 팀원의 보고를 경청한다.

■ **개발 및 테스팅에 관해 작업하면서 다음 사항들을 담당한다.**
- 기술 설계, UX 설계, 코딩 및 단위 테스팅
- 테스트의 설계, 수행, 평가 및 결함(defect) 해결
- 추가적인 정보가 필요한 경우, 제품 책임자 및 사용자와의 의사 소통
- 다른 팀원에 대한 코칭 및 지원

■ **인수 테스팅에 참여하면서 다음을 수행한다.**
- 인수 테스트 세트, 테스트 환경 또는 테스트 도구들을 준비한다.
- 제품 책임자 및 사용자가 인수 테스트를 수행하고 결과를 확인하도록 돕는다.
- 인수 테스팅에서 발생하는 결함 및 이슈를 처리한다.
- 인수 기준과 완료 정의가 충족되었는지에 대한 평가를 돕는다.

■ **배포(Deployment)에 참여해 다음을 수행한다.**
- 배포를 위한 빌드(builds)를 준비한다.
- 소프트웨어를 배포한다.
- 다른 팀들 및 IT 인프라 팀들과 연락한다.

- 사용자가 새 배포물에 익숙해졌는지 확인한다.

■ **회고 미팅에 참여해 다음을 수행한다.**

- 프로세스에 대한 개선 사항을 식별한다.
- 팀 리더와 합의한 대로 개선한다.

3.4.3 팀 구성원 작업 산출물

각 팀원의 활동으로 작업 산출물이 산출된다. 작업 산출물은 동작하는 소프트웨어를 인도하는 데 도움을 주는 실질적인 산출물이다. 팀원은 다음을 작업할 수 있다.

- 설계 문서
- 인도 가능한 코드
- 테스트 및 그 결과
- 배포 메뉴 및 배포되는 실행 파일과 같은 배포물(deployment artifacts)
- 사용자가 필요로 하는 보조 문서, 도움말 파일 또는 기타 배포물

3.4.4 팀원 기술 및 지식

팀원은 설계, 프로그래밍, 테스팅, 분석, UX, 배포 및 운영에 관한 기술을 보유하고 있을 수 있다. 각 팀원에게는 다양한 기술을 보유한 팀을 구성하기 위한 기술의 조합이 요구된다. 팀원은 팀에게 유익할 새로운 기술들을 익혀야 한다.

다른 팀원과 기술 공유

대부분의 팀원은 처음에 프로그래밍, 테스팅, 배포와 같은 특정 기술로 시작하지만 팀에서 이들은 곧 이터레이션 동안의 다양한 활동에 도움이 되기 위해서는 다양한 기술이 필요하다는 것을 알게 된다. 일반적으로 팀원은 여전히 한 영역에 특히 숙련된 것으로 간주되겠지만 이터레이션 전체 기간 동안 팀에 기여할 수 있는 새로운 기술도 습득할 것이다.

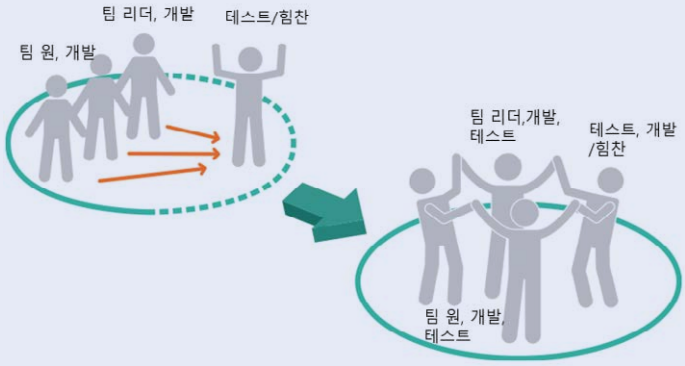
Tip 7. 하나 이상의 프로젝트에 대한 작업

이상적으로는 팀은 단 하나의 프로젝트에서 공동 작업한다. 하지만 팀원은 때로 2개 이상의 프로젝트에 대해 작업한다. 이 경우 팀원 모두는 팀 리더 및 다른 팀원과 공동 작업하면서 각 사람이 각 프로젝트에 대해 몇 시간을 할애하고 있는지 파악해야 한다.

예제 7. 희소한 자원인 한 팀원에 대한 운영

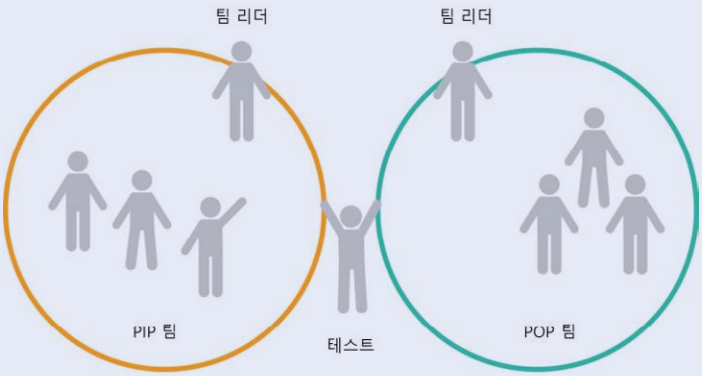
▶ '힘찬'은 단 하나의 프로젝트 '릴리(Lily)'에 대한 테스터로 일하고 있다.

다른 팀원은 개발자들이다. '힘찬'은 한 프로젝트의 테스터이다. 그는 유일한 테스터이다. 그는 4명의 개발자와 공동 작업하고 있다. 그가 모든 테스트를 수행할 수는 없다. 개발자들은 단위 테스팅 뿐만 아니라 스토리 테스팅도 익혀야 할 것이다. '힘찬'은 각 개발자와 차례로 짝 테스팅(pair testing)을 공동 수행해 이 일이 어떻게 수행되는지 보여준다. 그런 다음 개발자들은 서로의 스토리를 짝 테스트(pair test)해 준다. '릴리'는 게임 소프트웨어이므로 재미있을 것이다.



▶ '하늘'은 'PIP'와 'POP' 두 팀의 테스터로 일한다.

'하늘'은 테스터이다. 그는 'PIP' 팀과 'POP' 팀 모두에 대한 테스터로 지명되었다. 두 팀 모두 2명의 개발자와 1명의 팀 리더가 있다. '하늘'은 자신이 테스트할 시점에 대해 2명의 팀 리더 및 개발자들과 협의할 필요가 있다. 그는 또한 양 팀 모두의 일일 스탠드업 미팅에 참여할 방법을 모색해야 한다. 테스트 방식이 바뀔 것이다. 즉 개발자들이 스토리 테스팅을 보다 많이 담당해야 한다. '하늘'은 그들이 테스트를 효율적으로 수행할 수 있도록 코치하고 지원할 것이다.



3.5 전문가

전문가

■ 역할

팀에 대한 전문 기술의 일시적 제공

■ 설명

전문가들은 프로젝트와 팀의 필요에 따라 다양한 역할을 수행한다. 이들은 팀이 보유하지 않은 필요한 기술을 단기간 제공한다. 역할의 예로는 성능 테스터, 데이터베이스 전문가, 자동화 전문가 등이 있다.
전문가는 팀 리더에 의해 관리되며 다른 팀원 및 제품 책임자와 긴밀히 협력해 작업한다.

■ 활동

- 이터레이션 제로–4.2
- 이터레이션 계획–4.5
- 일일 스탠드업 미팅–4.6
- 개발–4.7
- 테스팅–4.8
- 인수 테스팅–4.9
- 회고–4.11

■ 작업 산출물

- 설계 문서
- 인도 가능한 코드
- 테스트 및 그 결과

3.5.1 역할

팀은 때로 영구적으로 필요하지는 않은 전문적 기술의 도움을 필요로 한다. 전문가는 팀이 동작하는 소프트웨어를 인도할 수 있도록 이러한 기술과 지식을 제공한다. 전문가들은 파트 타임 또는 단기로 팀에서 역할을 수행한다. 전문가는 필요한 경우에만 팀에 기술을 제공할 것이다. 각 파트 타임마다 한 번에 여러 팀을 위해 작업할 수도 있고 단기간 풀타임으로 한 팀에 파견될 수도 있다. 팀과 함께 작업하는 동안에는 팀 리더에게 보고할 것이다.

3.5.2 전문가 활동

팀의 일원으로 작업할 때 전문가들은 주로 자신의 전문 영역(예: 보안 테스팅) 내에서 작업하지만 여러 이터레이션을 수행하는 팀에 파견되는 경우에서처럼 때로는 필요에 따라 다른 분야의 작업도 수행한다. 전문가는 일일 스탠드업 미팅 및 회고 미팅과 같은 팀원이 수행하는 많은 활동에 참여할 것이다. 전문 분야와 관련된 특정 활동은 다음의 경우일 수 있다.

■ **이터레이션 제로 – 다음을 수행한다.**
- 프로젝트 계획서에 합의하는 데 도움이 되는 전문 영역의 지식을 제공한다.
- 시스템의 전반적인 아키텍처 구성에 도움이 되는 전문적 지식(예: 아키텍처, 성능, 보안 및 안정성 영역)을 제공한다.

■ **이터레이션 계획 – 다음을 수행한다.**
- 자신의 전문 영역인 사용자 스토리의 기능에 대한 조언(예: 추정, 의존성, 아키텍처 변경)을 제공한다.

■ **개발 및 테스팅 – 다음을 수행한다.**
- 자신의 전문 영역에 있는 개발(예: 보안 프로토콜 개발)을 수행한다.
- 자신의 전문 영역에 있는 테스팅(예: 접근성 테스팅)을 수행한다.

3.5.3 전문가 작업 산출물

전문가의 활동은 다른 팀원이 산출한 것과 같은 실질적인 작업 산출물을 자신의 전문 분야에 집중해 작성한다.

- **아키텍처 및 설계 (기술적 설계 및 UX 설계)**
- **인도 가능한 코드**
- **테스트 및 그 결과**

3.5.4 전문가 기술 및 지식

전문가에게는 팀이 현재 보유하고 있지 않은 전문 분야의 기술과 지식이 요구된다. 이 전문 분야는 모든 관련 IT 또는 도메인 영역일 수 있다.

Tip 8. 전문가가 팀 내에서 작업하는 방법

전문가는 단기간의 풀타임, 장기간의 파트 타임 또는 간헐적으로, 팀의 일원이 될 수 있다.

예제 8. 몇 차례의 이터레이션을 위해 팀과 단기간 공동 작업하는 전문가

▶ 단기간 풀타임으로 작업하는 전문가

'슬기'는 아키텍트(architect)이다. 그는 이터레이션 제로 및 처음 두 이터레이션 동안 'POP' 프로젝트 팀과 공동 작업해 해당 응용프로그램의 아키텍처가 합의돼 구현되도록 도움을 주었다. 그는 팀을 코치해 그들이 아키텍처와 설계에 대해 보다 많이 알 수 있도록 할 예정이다. 첫 두 번의 이터레이션에서 아키텍처 설계 및 기술적 설계에 도움을 준 '슬기'는 프로젝트를 떠나지만 팀이 도움을 필요로 한다면 언제든지 연락할 수 있게 한다.

예제 9. 이터레이션 동안 팀의 비정기적 작업하는 전문가

▶ 장기간 간헐적으로 활용되는 전문가

'경모'는 성능 튜닝 및 테스팅 전문가이다. 비정기적인 이터레이션을 위해 'PIP' 프로젝트 팀과 공동 작업을 수행한다. 그는 이터레이션 제로에 참여해 최적화된 성능의 아키텍처에 대한 조언을 했으며 이제 매 세 번째 이터레이션마다 팀을 방문해 그 때까지 인도된 모든 것과 해당 이터레이션에 대한 성능 테스트를 수행한다. 이처럼 성능 테스트와 튜닝을 수행하기 위해 간헐적으로 방문함으로써 팀은 성능에 관한 이슈들을 해결할 시간을 가질 수 있으며 피드백 루프(feedback loop) 사이에 너무 오랜 시간 간격을 두지 않아도 된다.

3.6 사용자

사용자

■ 역할

인도된 시스템의 사용

■ 설명

사용자는 인도된 시스템을 사용할 (또는 이미 사용하고 있는) 것으로 생각되는 이해 관계자이다. 사용자는 제품 책임자에게 요구 사항을 제공하면서 자신의 영역의 사용자 스토리에서 설명될 요구 사항에 대해 자세히 설명한다. 사용자는 또한 자신이 담당한 분야에 대한 인수 테스팅을 수행한다.
사용자는 제품 책임자 및 팀과 긴밀하게 협력한다.

■ 활동

- 새로운 기능 및 변경된 기능의 요구
- 사용자 스토리에 대한 상세한 설명
- 인수 테스팅-4.9

■ 작업 산출물

- 없음

3.6.1 역할

사용자는 새로운 시스템에 대한 요구 사항을 제공하고 구현된 소프트웨어가 이러한 요구 사항을 충족하는지 확인한다.
사용자는 인도된 시스템을 사용할 프로젝트의 이해 관계자들 중 어떤 사람이다. 사용자는 팀과 같은 조직에서 일하거나 다른 조직에서 일할 수도 있다. 일반적인 상용 소프트웨어의 경우 팀이 알 수 없는 많은 잠재적 사용자가 있을 것이다. 이 경우 사용자는 자신에게 소프트웨어를 제공 또는 판매하는 사람들로 대표될 수 있다.
사용자 또는 그 대리인은 제품 책임자 및 팀과 긴밀히 협력해 요구 사항을 제공하고 필요에 따라 사용자 스토리에 대한 추가 정보를 제공한다.
사용자는 인수 테스팅에 참여하고 배포된 후 동작하는 소프트웨어를 사용한다.

3.6.2 사용자 활동

사용자는 시스템의 요구 사항을 식별해 주고 해당 사용자 스토리를 상세하게 설명하며 인수 테스팅을 수행해 피드백을 제공한다.
프로젝트 사용자는 다음을 수행한다.

- 제품 책임자가 제품 백로그에 사용자 스토리로 기록할 변경 사항 및 새로운 기능에 대해 제품 책임자에게 요구한다.
- 팀의 질문에 답하고 요구 사항을 명확히 하며 사용자 스토리에 대해 상세하게 설명한다.
- 자신이 요구한 기능에 대한 인수 테스팅에 참여한다.

3.6.3 사용자 작업 산출물

사용자에게는 특정 작업 산출물의 개발을 위해 요구되는 사항이 없다. 제품 책임자와의 관계에 따라 새롭거나 변경된 기능에 대한 요구를 문서화하기로 결정할 수 있지만 이것은 일반적으로 비공식 처리된다.

3.6.4 사용자 기술 및 지식

프로젝트 사용자는 제품 책임자에게 새롭거나 변경된 기능에 대해 요구할 수 있도록 사업 영역에 관한 지식을 갖추어야 한다.

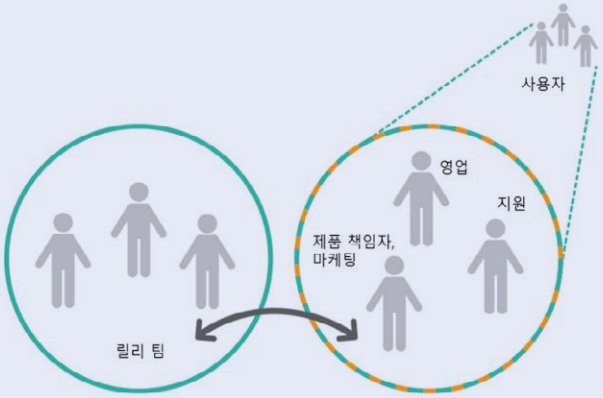
Tip 9. 사용자와 접촉할 수 없는 경우

동일한 조직 내에서 근무하는 동료를 위한 소프트웨어를 구축하고 있을 경우 사용자를 만나는 일은 쉽다. 다른 조직의 사용자를 위한 소프트웨어를 제작하고 있을 경우에도 사용자를 방문해 대화할 수 있다. 그러나 때로는 사용자와 접촉할 수 없는 경우도 있다. 예를 들어 사용자가 상용 소프트웨어 패키지를 구매할 예정이거나 (혹은 이미 구매했거나) 웹사이트를 이용하는 대중의 일원일 수도 있다. 이러한 경우 직접 대화할 수 없기 때문에 그들을 대표해서 새롭거나 변경된 기능을 제안하고 인수 테스팅에 참여해 줄 사람을 찾아야 한다. 때로 사용자는 영업, 마케팅 및 지원 팀의 대표일 수 있다.

예제 10. 상용 소프트웨어를 제작하는 경우의 사용자 대표

▶ 최종 사용자가 대중의 일원이다.

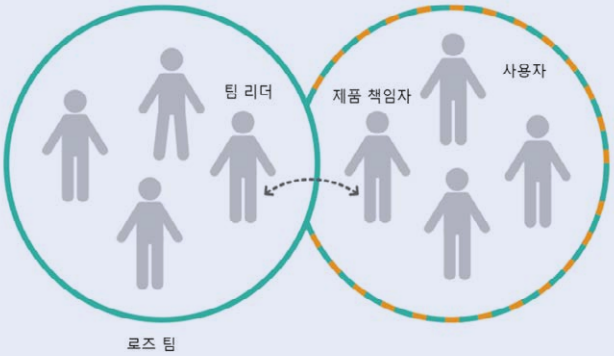
'릴리(Lily)' 프로젝트 팀은 상용 (게임) 소프트웨어를 제작하고 있다. 사용자는 해당 소프트웨어를 취급할 영업, 마케팅 및 지원 팀이 대표한다. 제품 책임자는 마케팅 팀의 구성원일 것이다.



예제 11. 사내 소프트웨어를 제작하면서 사용자와 직접 대화할 수 있는 경우

▶ 최종 사용자와 접촉할 수 있다.

'로즈(Rose)' 프로젝트 팀은 사내에서 소프트웨어 시스템을 사용할 사용자를 위한 소프트웨어를 제공할 예정이다. 이들은 최종 사용자와 대화할 수 있으며 해당 프로젝트의 제품 책임자는 고객 회사의 직원이다.



3.7 고 객

고 객

■ 역할

인도될 시스템에 대한 비용 지불

■ 설명

고객은 인도될 시스템의 상위 목표를 결정한다. 또한 예산 및 납기와 같은 개발에 대한 높은 수준의 제약사항도 결정한다.
고객은 제품 책임자와 긴밀하게 협력한다.

■ 활동

• 이터레이션 제로-4.2

■ 작업 산출물

• 프로젝트 계획서-6.1

3.7.1 역할

고객은 프로젝트의 비용을 지불하는 사람, 집단 또는 조직이다. 고객은 인도받을 시스템에 대한 비용을 지불하며, 프로젝트에 대한 목표와 정보를 제공하고 프로젝트 완료 여부를 승인한다.

3.7.2 고객 활동

고객은 프로젝트의 방향을 수립하고 비용을 지불한다. 고객은 이터레이션 제로 동안 제품 책임자 및 팀 리더와 긴밀하게 협력해 프로젝트 목표가 프로젝트 계획서에 명확하게 정의되도록 할 것이다. 또한 예산, 납기 및 기타 제약 조건들이 명확하게 파악될 수 있도록 할 것이다. 프로젝트 기간 동안 제품 책임자의 문의 사항이 있을 경우 접촉 가능할 것이며 진행에 방해가 되는 사항의 해소에 영향을 줄 수 있는 경우 도움을 줄 것이다.

3.7.3 고객 작업 산출물

고객은 프로젝트 요구 사항을 모든 사람이 파악할 수 있도록 할 책임이 있다. 고객은 프로젝트 목표에 관한 한 페이지 분량의 간략한 설명을 프로젝트 계획서 초안의 형식으로 제공해야 한다. 이 프로젝트 계획서 초안은 이터레이션 제로 동안 완성돼 합의된다.

3.7.4 고객 기술 및 지식

고객은 사업 및 사업 영역에 관한 지식이 필요하다. 고객은 프로젝트에 대한 전략적 목표, 기대 이익 및 지출할 준비가 된 비용 또는 예산을 파악해야 한다. 프로젝트를 완성하거나 완성하지 못하는 데 따른 조직의 리스크를 파악해야 한다.

Tip 10. 바쁜 고객 – 이터레이션 제로

제품 책임자의 관점에서 볼 때 고객이 이터레이션 제로에 참여해야 하겠지만 고객이 직접 참여할 수 없는 경우라도 선택할 방법은 있다. 즉 고객과의 미팅을 연기하거나 전화나 화상 회의로 고객을 참여시키거나 고객이 시간 여유가 있을 때 인터뷰를 통해 프로젝트 계획서 초안에 담길 목표와 제약 조건 등에 대해 팀 리더와 논의하게 하는 것이다.

예제 12. 이터레이션 제로, 바쁜 고객 및 프로젝트 파악

▶ 프로젝트 '로즈(Rose)를 위해 제품 책임자 '경택'이 고객 '나영'과 인터뷰한다.

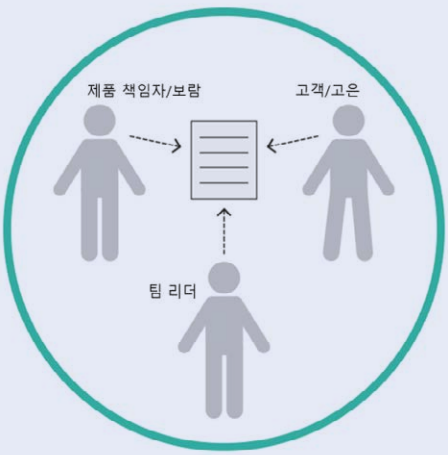
'경택'의 고객인 '나영'은 이터레이션 제로 오프닝 미팅에 참석할 시간이 없지만 맞추어야 할 마감 시간이 있기 때문에 프로젝트가 제 때 시작되는 것이 중요하다. '경택'은 제품 책임자로서 자신이 프로젝트 목표를 파악하고 있는지를 확인하기 위해 '나영'과 미팅을 가진다. 목표가 문서화되지 않았기 때문에 그는 이터레이션 제로 첫 미팅에 메모를 가져간다. 팀 리더가 임명되었을 때 그는 목표에 대해 질문하고 몇몇 이슈와 리스크를 제시한다.

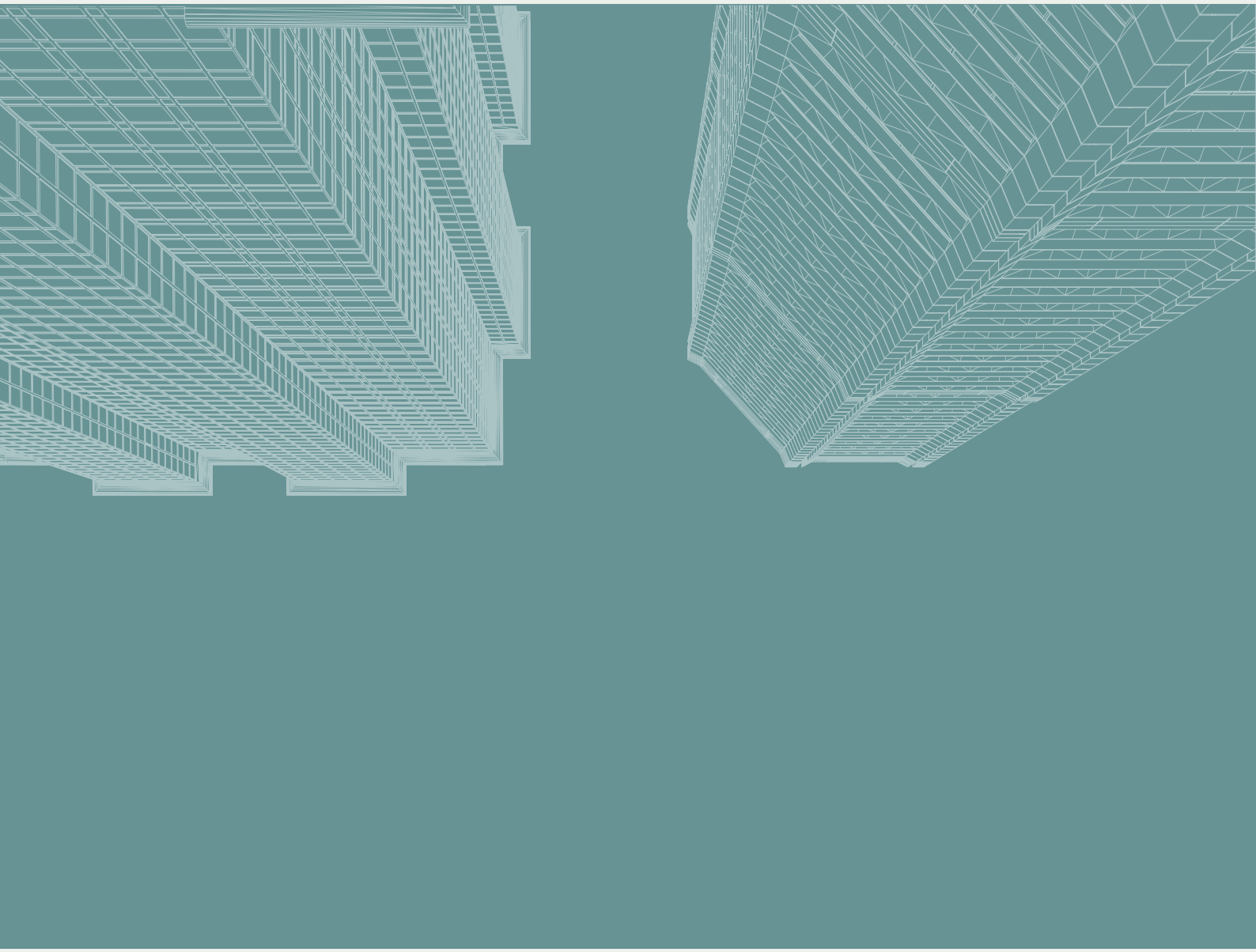
'경택'은 고객과의 두 번째 미팅을 마련해 팀 리더와 함께 참석한다. 이들은 프로젝트 계획서의 역할을 할 고객의 목표, 제약 조건 및 리스크에 대한 마인드맵을 작성한다.

▶ 제품 책임자 '보람'이 프로젝트 '릴리(Lily)' 미팅을 연기한다.

'보람'의 고객 '고은'은 '릴리-고우즈' 게임 회사의 영업 및 마케팅 책임자이지만 이터레이션 제로 오프닝 미팅에 참석할 시간이 없다. 프로젝트 목표가 명확하지 않다. '보람'은 고객과 접촉할 수 있을 때까지 오프닝 미팅을 연기할 것을 제안한다.

이것은 이터레이션 제로가 일주일간 연기됨을 의미하지만 고객이 미팅에 참석하게 될 경우 '보람'은 팀의 팀 리더와 함께 필요한 것을 명확하게 공유해 파악할 수 있고, 고객이 한 페이지 분량의 프로젝트 계획서 초안을 매우 신속하게 작성하도록 도울 수 있다.





CHAPTER 04

프로젝트 수행 개요

CHAPTER 04

프로젝트 수행 개요

4.1 프로젝트 계획 수립

프로젝트 계획 수립

■ 목적

프로젝트를 운영하는 이유 파악하기

■ 설명

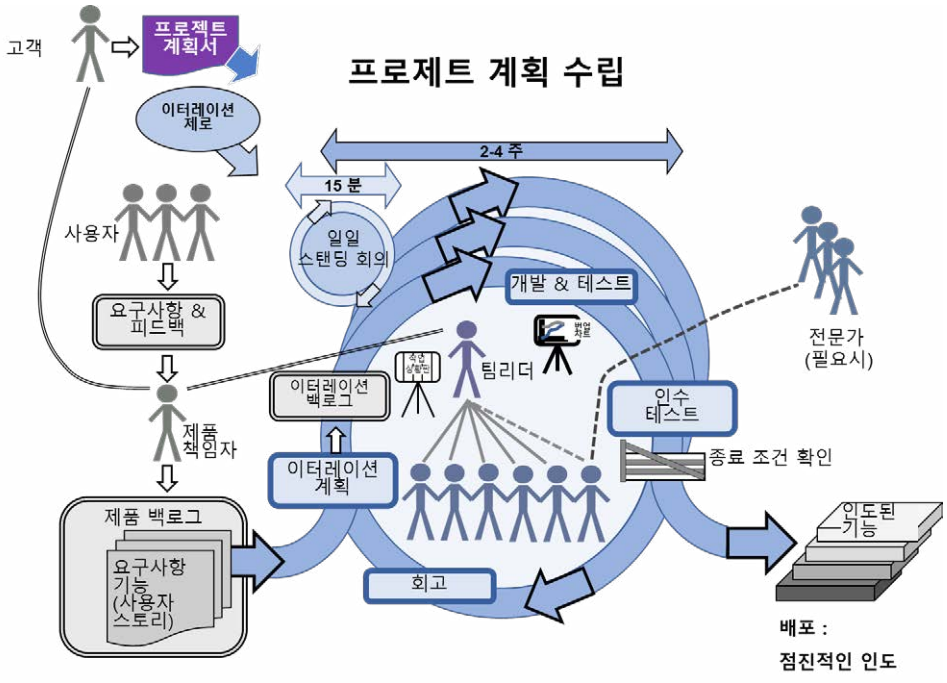
고객은 목표, 제약 조건 및 성공 지표를 포함해 프로젝트가 필요한 이유를 식별하고 이를 프로젝트 이해 관계자와 합의한다.

■ 작업 산출물

- 프로젝트 계획서 초안–6.1

■ 역할

- 고객–3.7
- 제품 책임자–3.2
- 팀 리더–3.3



〈 그림 2. 구조도에서의 프로젝트 계획 〉

프로젝트는 고객, 사용자 및 프로젝트 목표를 아는 경우에 시작될 수 있다.

고객이 다른 프로젝트보다 우선 순위로 정했기 때문에 해당 프로젝트를 시작한다. 일반적으로 해당 프로젝트 완료 시의 상대적 이익, 관련된 프로젝트 리스크 및 비용에 기초해 결정이 내려진다.

- 고객이 프로젝트가 필요하다고 결정했다면 그 비용을 지불하겠다는 의미이다.

모두는 프로젝트의 소프트웨어를 누가 사용할 지 알고 있다. 누구에게 이익이 될 지, 그들의 업무 영역이 무엇인지, 해당 소프트웨어가 어떻게 도움이 될 지 분명히 알고 있다.

- 해당 소프트웨어의 사용자(또는 대표자)가 식별된 후 프로젝트가 진행되면서 시스템 요구 사항을 확인해 줄 것을 사용자에게 요구한다.

고객이 프로젝트에서 기대하는 것인 예산 및 납기에 대해 모두들 명확히 알아야 한다. 고객이 프로젝트 목표(목표, 기대 이익, 리스크, 제약 조건, 성공 지표)를 수립하고 제품 책임자 및 팀 리더가 이에 동의한다.

- 프로젝트 목표(달성하고자 하는 것, 관련된 기회 및 리스크)가 합의된다.
- 가장 중요한 제약 조건(예산과 납기)이 합의된다.

프로젝트 시작 전에 고객이 프로젝트 목표를 결정한다. 고객은 그 목표를 프로젝트로 가져오고, 제품 책임자 및 팀 리더와 함께 그 목표를 프로젝트 계획서로 합의한다. 이것은 **이터레이션 제로**의 일환으로 행해진다.

4.1.1 프로젝트 목표

고객은 달성하고자 하는 것에 대한 최초의 비전을 갖고 있으며 이는 종종 사용자 요구에 의해 유도된다. 프로젝트 목표는 고객이 최종적으로 인도받고 싶어하는 것이다. 하지만 이러한 프로젝트 목표가 처음부터 완전하지 않을 수 있다. 고객과 사용자는 초기 단계에서 자신들이 원하는 모든 것을 알지 못할 수 있으며, 필요한 세부 사항을 제공하지 못할 수 있다. 또한 프로젝트가 진행됨에 따라 상황이 바뀔 것이므로 고객이 원하는 것도 변경될 것이다.

프로젝트를 시작하는 이유는 다음과 같다.

- 고객이 새로운 기회를 발견하고 그 기회를 이용하고자 한다.
- 사용자가 해결해야 할 문제가 있다.
- 조직 외부로부터의 법적, 규제적 또는 기타의 압력들이 변경을 요구한다.
- 고객이 반드시 수용해야 하는 새로운 리스크가 발생했을 수도 있다.

프로젝트는 새로운 시스템의 구축, 현 시스템의 업데이트 혹은 현 시스템의 교체 등이 있을 수 있다.

4.1.2 기대 이익

고객은 프로젝트가 새로운 활동(예: 신규 고객 확보, 신제품 인도, 새로운 시장 진출 및 지역 사회 공헌)를 지원해 주기를 바라거나 현 시스템이 새로 제정된 법률을 충족(예: 곧 있을 금융 규제를 충족)시킬 수 있음을 보장 받고 싶어 한다. 성공적인 프로젝트로부터 기대되는 이익(예: 수익 증대, 고객 기반 확대, 규제적 벌금 가능성의 배제 등)이 무엇인지 명확하게 알아야 한다.

4.1.3 리스크 관리

프로젝트 기간 동안 리스크가 다루어져야 한다. 리스크를 식별하고, 평가하고 관리하기 위해 일반적인 리스크 관리 프로세스를 사용하라.

아래 사항에 대한 리스크를 알고 있어야 한다.

■ **조직의 경우**

- 전문 기술과 인력의 부족
- 의사 소통 문제
- 협력 업체가 역할 수행에 실패

■ **프로젝트의 경우**

- 불충분한 자원
- 필요한 테스트 환경 구축 일정 지연
- 기술적 제약을 고려하지 않은 요구 사항

■ **소프트웨어의 경우**

- 장애가 나기 쉬운 소프트웨어의 출시 가능성
- 취약한 소프트웨어 품질 특성
- 취약한 데이터 무결성 및 품질

4.1.4 이슈 관리

프로젝트 기간 동안 이슈가 다루어져야 한다. 이슈는 발생되지 않았을 때에는 리스크였지만, 발생되면 처리해야 한다. 이슈는 조직, 프로젝트, 팀 또는 산출물의 문제일 수 있다.
이슈는 일반적인 프로젝트 활동 동안 다루어진다. 이러한 활동 동안 해결될 수 없는 경우 팀 리더 또는 제품 책임자가 그 이슈를 담당하게 된다.

■ **이터레이션 진행에 영향을 미치는 이슈(방해 사항)**
- 일일 스탠드업 미팅 중에 제기된다.
- 미팅이 끝난 직후에 다루어진다.
- 필요한 경우 조치 목록에 기록된다.

■ **프로젝트를 위한 활동 및 작업 산출물에 영향을 미치는 (잠재적 개선 사항) 이슈**
- 회고 미팅을 통해 제기돼 다루어진다.
- 회고 미팅에서의 조치로 기록된다.

■ **인도 가능한 산출물(예: 소프트웨어)에 영향을 미치는 (결함(defects)) 이슈**
- 스토리 또는 인수 테스팅 동안 **비공식적 결함 관리(Informal Defect Management, 6.12.4 참고)** 의 일부로 해당 이터레이션에 제기돼 다루어진다.
- 해당 이터레이션 동안 해결되거나 제품 백로그에 추가된다.
- 운영 및 유지보수 기간 동안 제기되는 경우 제품 백로그의 일부로 제기돼 다루어진다.

4.1.5 제약 조건

프로젝트 제약이 무엇인지 분명히 알아야 한다. 여기에는 다음이 포함될 수 있다.
- 고객이 지불할 준비가 된 금액: 예산은 얼마인가?
- 고객이 기다릴 준비가 된 시간: 납기는 언제인가?
- 법적 및 규제적 제약: 반드시 해야 하는 것은 무엇이며 우리에게 허락된 것은 무엇인가?

4.1.6 성공 지표

프로젝트의 성공을 어떻게 측정할 지 명확하게 알아야 한다. 성공 지표는 목표, 기대 이익, 리스크 및 제약과 같은 목표들과 관련돼 있다.

예시:
- 정성적 지표– 예: 새로운 시스템에 대한 사용자의 만족도
- 정량적 지표– 예: 소프트웨어가 출시된 후 달성된 판매량의 측정

지표는 소프트웨어를 인도한 후에 측정되며 이 새 소프트웨어가 어떤 차이를 나타냈는지 식별해 준다.

4.1.7 프로젝트 목표에 동의하기

프로젝트 목표는 주요 프로젝트 이해 관계자들(고객, 제품 책임자 및 팀 리더)이 동의해야 한다. 이러한 동의는 **이터레이션 제로**의 일환으로 발생한다. **프로젝트 계획서** 초안은 고객이 이터레이션 제로에 가져와서 이터레이션 제로 기간 동안 동의를 얻어야 한다.

4.1.8 프로젝트 목표 기록하기

프로젝트 기간 동안 전체 목표를 놓칠 수 있다. 목표는 처음에 이터레이션 제로 동안 아주 간단하고 명확하게 문서화돼야 하며 프로젝트가 진행되면서 업데이트돼야 한다. 일반적으로 사용되는 프로젝트 계획서는 팀 영역에서 쉽게 볼 수 있는 위치에 배치돼야 하며 다음의 형태로 작성될 수 있다.

- 마인드 맵
- 포스터 또는 배너
- 기타 – 짧고 파악하기 쉬운 형식

4.1.9 프로젝트와 프로그램

많은 조직에서 프로젝트는 프로그램의 일부로 발생하며 그 중 일부는 소프트웨어 프로젝트이고 일부는 다른 유형의 프로젝트이다. 프로젝트가 보다 큰 업무 프로그램의 일부인 경우 팀 리더 및 제품 책임자는 프로젝트 사이의 관계 및 의존성을 인지할 필요가 있다.

예제 13. 프로그램 내의 프로젝트: 오키드딜라이츠

▶ 오키드딜라이츠(OrchidDelights): 프로그램내 프로젝트

아름씨는 난초(orchid)를 재배하는 종묘원 담당 임원이다. 그녀는 종묘원에서 생산, 판매, 마케팅, 재무 및 행정을 책임지고 있다. 그녀의 회사가 난초를 재배하고, 수확하고, 판매하고, 배달하는 모든 것을 담당한다.

그녀는 '오키드딜라이츠'라 불리는 업무 수행 개선 프로젝트의 고객이다. 그녀는 난초를 주문하고 대금을 지불할 수 있는 웹사이트를 구축하기 위해 프로젝트에 참여하고 있다. 그녀는 백오피스 시스템을 개선해 새로운 웹사이트와 통합하기 위한 프로젝트에 참여하고 있다.

그녀는 또한 새로운 웹사이트를 론칭하기 위한 영업 마케팅 캠페인과 종묘원의 관개 제어시스템을 개선시키기 위한 임베디드 소프트웨어 프로젝트에도 참여하고 있다. 웹사이트 구축과 백오피스 시스템을 위한 팀 리더들과 제품 책임자들은 소프트웨어 통합을 위해 협업해야 한다.

웹사이트 구축을 담당한 팀 리더와 제품 책임자는 영업 및 마케팅 론칭 캠페인 프로젝트와 연락을 취해 웹사이트 론칭 준비가 마케팅 캠페인 날짜들과 맞는지 확인해야 한다.

4.2 이터레이션 제로

이터레이션 제로

■ 목적

프로젝트 준비 및 초기 릴리즈 계획

■ 설명

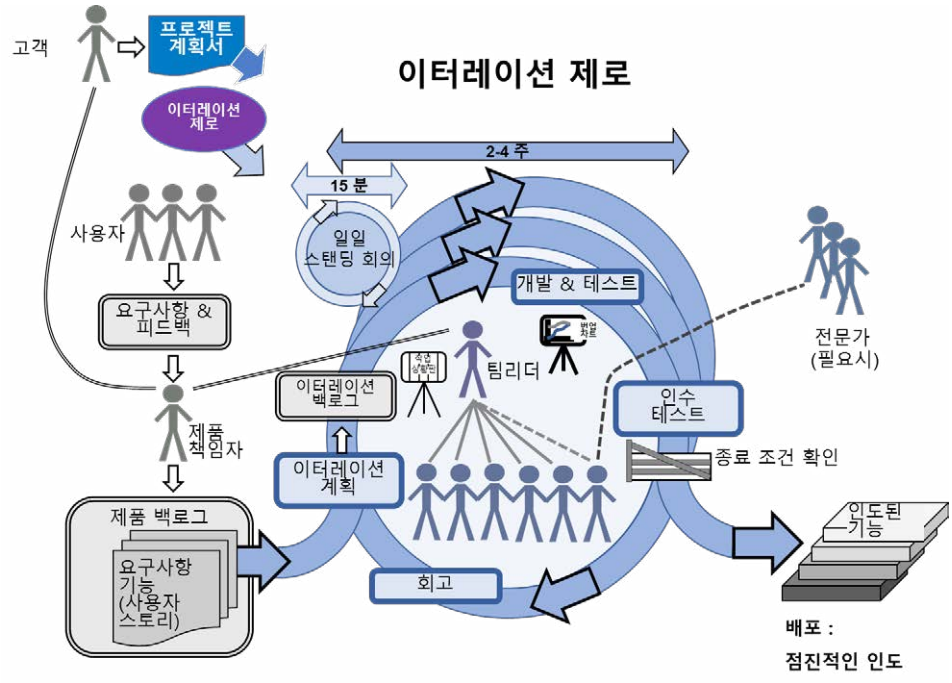
첫 번째 이터레이션이 수행되는 데 필요한 모든 것이 이터레이션 제로에 준비된다. 처음에 이것은 프로젝트 계획서에 대해 고객과 합의하는 것을 의미한다. 개발 프로세스가 시스템 아키텍처와 함께 정의된다. 최초 릴리즈 계획이 합의될 것이고 충분한 사용자 스토리가 첫 번째 이터레이션을 위해 작성될 것이다. 프로젝트에 참여할 모든 사람과 지원 인프라가 준비돼야 한다.

■ 작업 산출물

- 프로젝트 계획서-6.1
- 릴리즈 계획-6.3
- 릴리즈 번-업 차트-6.4
- 아키텍처 설계(Architectural Design)
- 제품 백로그 (사용자 스토리)-6.6
- 작업상황판-6.8

■ 역할

- 고객-3.7
- 제품 책임자-3.2
- 팀 (팀 리더 및 팀원)-3.1
- 전문가-3.5



〈 그림 3. 구조도에서의 이터레이션 제로 〉

4.2.1 이터레이션 제로 시기

이터레이션 제로는 프로젝트가 시작될 때 한 번 발생한다.

4.2.2 이터레이션 제로가 무엇인가?

일단 고객이 프로젝트를 승인하면 수행돼야 할 몇몇 준비 활동이 있다. 첫 작업 이터레이션을 시작하기 전에 이터레이션 제로에서 그러한 것들이 준비된다. 이러한 활동에는 다음이 포함된다.

- 제품 책임자, 팀 리더 및 올바른 자세, 기술 및 경험을 갖춘 팀을 모집해 모든 사람들이 준비되도록 한다.
- 프로젝트 계획서의 형태인 프로젝트 목표에 합의한다.
- 초기 시스템 아키텍처를 정의한다.
- 초기 개발 프로세스와 '완료 정의(Definition of Done)'를 정의한다.
- 이 프로젝트에 대한 '스토리 포인트'를 정의하고 팀의 예상 속도에 대한 초기 평가를 함으로써 이터레이션 계획을 준비한다.
- 프로젝트를 위한 인프라 및 자원을 확보한다.
- 최초 릴리즈 계획에 합의한다.
- 제품 백로그가 첫 번째 이터레이션을 위한 충분하고 완전한 사용자 스토리를 갖추고 있는지 확인한다.

4.2.3 이터레이션 제로 역할

이터레이션 제로 참여자들의 역할은 다음과 같다.

■ **고객은 다음을 수행한다.**

- 제품 책임자의 지명에 동의한다.
- 프로젝트 계획서의 형태로 프로젝트 목표(목표 및 제약 사항)를 제공한다. 이것은 제품 책임자 및 팀 리더가 동의해야 한다.

■ **IT 팀의 의사 결정자는 다음을 수행한다.**

- 팀 리더를 지명한다.
- 팀원을 지명한다.

■ **제품 책임자는 다음을 수행한다.**

- 프로젝트 계획서에 대해 고객 및 팀 리더와 합의한다.
- (사용자 스토리 작성을 포함해) 초기의 제품 백로그 관리를 수행한다.
- 팀 리더의 도움을 받아 초기의 릴리즈 계획을 수립한다.
- 개발 프로세스 및 완료 정의에 대해 팀 리더와 합의한다.

■ **팀 리더는 다음을 수행한다.**

- 프로젝트 계획서에 대해 고객 및 제품 책임자와 합의한다.
- IT 팀의 의사 결정자를 도와서, 팀원을 확보한다.
- 팀에 필요할 인프라와 자원을 식별해 확보한다.
- 초기의 시스템 아키텍처를 정의한다.
- 제품 책임자를 도와서 초기의 릴리즈 계획을 수립한다.
- 개발 프로세스와 완료 정의에 대해 제품 책임자와 합의한다.

■ **팀원은 다음을 수행한다.**

- 초기의 시스템 아키텍처, 개발, 배포 프로세스 및 완료 정의를 숙지한다.
- 이 프로젝트에 대한 '스토리 포인트'의 정의와 팀의 속도 측정을 위한 시작점에 동의한다.
- 필요한 경우 팀 리더를 돕는다.

■ **전문가는 다음을 수행한다.**

- 전문 분야의 지식으로 프로젝트 계획서 작성을 지원한다.
- 초기의 시스템 아키텍처를 정의하는 데 있어서 팀 리더를 지원한다.
- 필요한 경우 팀 리더를 돕는다.

4.2.4 이터레이션 제로 작업 산출물

이터레이션 제로 동안 다음의 작업 산출물을 작성한다.

- 프로젝트 계획서
- 최초 릴리즈 계획
- 초기의 제품 백로그
- 초기의 시스템 아키텍처

이러한 작업 산출물을 작성하기 위해 이터레이션 제로 활동에는 다음의 작업들이 포함된다.

1. 이터레이션 제로 스타트업 미팅: 고객과 IT 의사 결정자가 만나서 프로젝트가 수행될 것에 합의한다. 이들은 제품 책임자와 팀 리더를 지명한다.
2. 프로젝트 계획서: 고객, 제품 책임자 및 팀 리더가 만나서 프로젝트 계획서에 대해 합의한다.
3. 모집(Recruitment): 팀 리더는 팀원 및 모든 전문가를 확인해 요구하고 지명한다. 제품 책임자는 사용자를 확인한다.
4. 자원 확보(Resourcing): 팀 리더는 필요할 인프라와 자원을 식별해 요구하며 팀은 이것들이 사용될 준비가 되었는지 확인하고 테스트한다.

a. 팀 작업 공간, 데스크 및 장비
b. IT 인프라, 도구, 환경
c. 보안 (IT, 장비, 정보 액세스)
d. 기타 필요한 자원

5. 시스템 아키텍처: 팀 리더와 팀(필요한 경우 전문가)은 초기의 시스템 아키텍처를 설계하고 검토한다.
6. 제품 백로그 관리 및 릴리즈 계획: 제품 책임자는 팀 리더의 도움을 받아 초기의 제품 백로그 및 최초 릴리즈 계획을 작성한다.
7. 초기의 프로세스 정의 및 완료 정의: 팀 리더, 팀원 및 제품 책임자는 프로세스와 완료 정의에 대해 합의한다.
8. 이터레이션 계획 동안의 추정을 위한 준비: 프로젝트와 팀을 위한 스토리 포인트의 정의 및 이터레이션 동안의 예상 작업 속도의 초기 평가
9. 프로젝트 시작 미팅: 이터레이션 제로 팀 전체가 만나서 프로젝트 이터레이션 1의 시작 준비에 동의한다. 모든 사람이 목표에 대해 약속한다. 고객은 시작 준비를 승인한다.

Tip 11. 타임박싱(Timeboxing) 이터레이션 제로

시작하기 위한 필수적인 활동으로 수행되기 위해 이터레이션 제로의 시간을 한정하는 것은 가치 있는 일이다. 매 이터레이션의 끝에는 회고 활동이 있음을 잊지 마라. 이를 통해 팀이 프로세스와 활동을 개선할 것으로 기대되기 때문이다. 지금 모든 것을 적소에 두려고 하지 마라. 작게 시작해 매 이터레이션을 지나면서 작업 방법, 작업 환경 및 팀이 점진적이며(incrementally) 반복적으로(iteratively) 변경될 것이라고 기대하라.

Tip 12. 시스템 제품군

프로젝트가 시스템 제품군 중의 하나이고 동일한 네트워크 상에서 호스팅되며 동일한 팀에 의해 구축되고 있는 경우 이미 만들어진 대부분의 결과와 결정으로 인해 이터레이션 제로는 극히 단기간의 활동이 된다.

4.3 제품 백로그 도출 및 관리

제품 백로그 도출 및 관리

■ 목적

충분한 고품질의 사용자 스토리들이 제품 백로그에서 이용 가능함을 보장

■ 설명

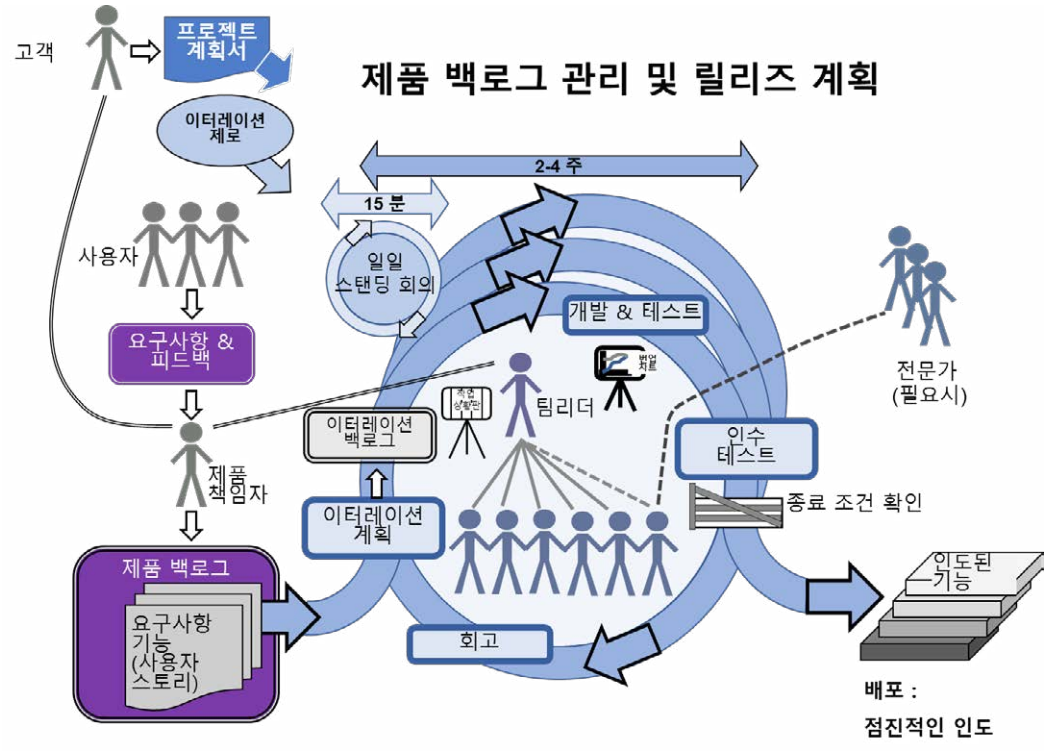
이터레이션 계획을 수립하기 전에 제품 백로그에 충분한 사용자 스토리들을 확보해, 적절한 부분집합이 다음 이터레이션에서의 구현을 위해 선택될 수 있도록 한다. 이 활동은 릴리즈 계획의 제약 조건들을 충족시키는 완전한 사용자 스토리들을 제품 백로그가 충분히 확보하고 있음을 보장한다.

■ 작업 산출물

- 제품 백로그 (사용자 스토리)–6.6

■ 역할

- 제품 책임자–3.2



〈 그림 4. 구조도에서의 제품 백로그 관리 〉

4.3.1 제품 백로그 관리 시기

제품 백로그 관리는 이터레이션 제로 동안 처음 수행돼 첫 번째 이터레이션을 위한 사용자 스토리들이 충분히 있는 지 확인한다. 이후 빈번하게 수행되는데 처음에는 사용자 및 팀의 최신 요구가 제품 백로그에 포함돼 있는지 확인하고, 다음으로 이터레이션 계획을 지원하기 위해 제품 백로그에 충분한 사용자 스토리들이 있는 지 확인하며, 세 번째로 프로젝트가 고객 및 사용자의 목표와 계속 일치하도록 보장한다.

4.3.2 제품 백로그 관리가 무엇인가?

이터레이션 계획을 수립하기 전에 제품 백로그에 충분한 사용자 스토리들을 확보해 우선 순위가 높은 사용자 스토리들이 다음 이터레이션에서 구현을 위해 선택될 수 있도록 한다. 제품 백로그 관리란 릴리즈 계획의 제약 조건들을 완전히 충족시키는 사용자 스토리들을 제품 백로그가 충분히 확보하고 있음을 보장하는 것이다.

제품 책임자는 다음을 수행한다.

- 사용자와 자주 의사 소통해 사용자가 새로운 기능과 변경된 기능을 요구할 수 있게 한다.
- 팀과 자주 의사 소통하면서 기술적 요구를 한다.
- 인수 기준 및 우선 순위를 포함해 요구에 대한 사용자 스토리를 작성한다.
- 필요한 경우 기술적 요구인 사용자 스토리의 작성을 팀에 위임한다.
- 팀 리더 및 사용자와 함께 사용자 스토리를 검토해 이것이 정확하면서도 (구현될 만큼 충분히 상세하게) 준비돼 있는지 확인한다.
- 다음 이터레이션을 위해 항상 충분한 사용자 스토리들이 준비돼 있는지 확인한다.

또한 고객에게 연락해 프로젝트 목표가 고객의 목표와 일치하는지 확인한다. 때로 고객을 위한 목표가 프로젝트 기간 중에 변경되는 데 이는 해당 프로젝트 백로그의 우선 순위가 다시 정해져야 할 수도 있음을 의미한다. 프로젝트 계획서를 업데이트해야 할 경우 제품 책임자는 팀 리더의 동의를 얻어야 한다. 팀 리더는 프로젝트 계획서를 업데이트해 이를 팀과 공유한다.

4.3.3 제품 백로그 관리 역할

제품 백로그 관리는 사용자로부터 의견을 수렴해야 하는 제품 책임자와 팀이 담당한다. 제품 책임자는 제품 백로그가 최신 상태이며 다음 이터레이션 계획 미팅을 위해 준비돼 있는지 확인해야 한다.

Tip 13. 스토리 카드를 사용하는 이유

각 스토리마다 하나씩의 A6 사이즈 스토리 카드를 사용하면 매우 유용한다. 이것은 이터레이션 백로그와 해당 이터레이션의 진행 상황을 모든 사람이 명확히 볼 수 있게 하기 때문에 매우 중요하다. 카드 그룹(pile)의 사이즈는 이터레이션 백로그의 사이즈를 말해 준다. 카드 작업이 진행됨에 따라 카드들은 작업상황판에서 물리적으로 이동한다. 수행돼야 할 일, 그 일의 담당자 및 진행 상황을 모든 사람이 알 수 있다.

4.4 릴리즈 계획

릴리즈 계획

■ 목적

릴리즈 스케줄 및 내용의 결정

■ 설명

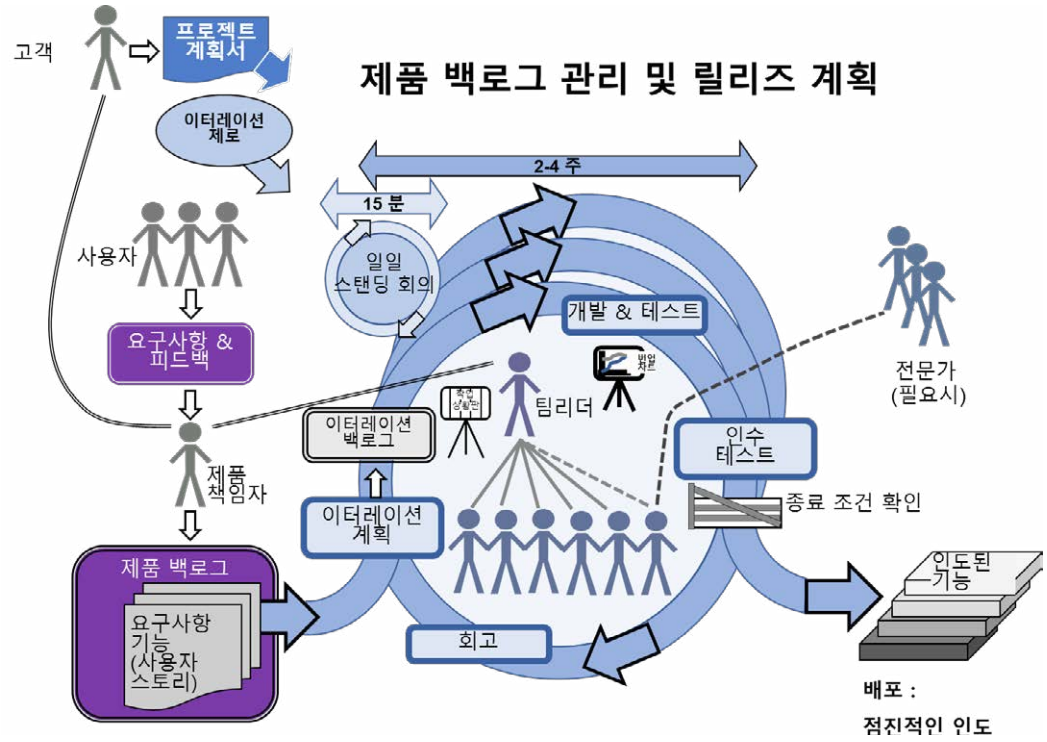
프로젝트가 진행됨에 따라, 최초 릴리즈 계획에 대한 변경 및 확장이 필요할 수 있다. 향후 릴리즈의 내용은 고객 및 사용자 요구사항뿐만 아니라 기술적 제약 사항 및 의존성을 기반으로 결정된다.

■ 작업 산출물

- 릴리즈 계획서-6.3
- 번-업 차트-6.4

■ 역할

- 고객-3.7
- 제품 책임자-3.2
- 팀 리더-3.3
- 사용자-3.6
- 전문가-3.5



〈 그림 5. 구조도에서의 릴리즈 계획 〉

4.4.1 릴리즈 계획 시기

릴리즈 계획은 처음에는 이터레이션 제로 동안에 수행되고, 이 후 필요에 따른 간격으로 수행된다. 릴리즈 계획은 제품 백로그 관리와 연계될 수 있다.

4.4.2 릴리즈 계획은 무엇인가?

릴리즈 계획은 소프트웨어를 배포하는 빈도와 각 릴리즈의 내용 또는 목표를 결정하는 활동이다. 어떤 프로젝트에서는 매 이터레이션의 완료 시에 사용자에게 동작하는 소프트웨어를 배포한다. 다른 프로젝트에서는 보다 긴 간격으로 소프트웨어를 사용자에게 릴리즈한다. 릴리즈 계획 참여자는 다음을 수행한다.

- 릴리즈가 요구되는 빈도 식별
- 각 릴리즈의 목표 식별
- 그러한 목표를 달성하기 위해 필요한 사항 식별

각 릴리즈의 내용에는 사용자 기능, 기술적 기능 및 변경 사항이 조합돼 있다. 해당 릴리즈의 목표는 다음 중 하나일 수 있다.

- 비즈니스/도메인 이벤트(예: 제품 전시회) 또는 규제/법률 이벤트(예: 새로운 법률의 시행)
- 사용자가 선호하는 소프트웨어 업데이트 방식
- 이슈에 대한 긴급 수정
- 하나의 이터레이션에 완성될 수 없는 작업
- 조직의 경영진, IT 인프라 팀 또는 운영 팀이 선호하는 소프트웨어 업데이트 방식

4.4.3 릴리즈 계획 역할

릴리즈 계획 역할에는 다음이 포함된다.

- **고객:**
  - 프로젝트 목표에 기반해 릴리즈의 비전 또는 목표 제공
- **제품 책임자:**
  - 우선 순위에 기반해 릴리즈 계획 수립
  - 해당 릴리즈 계획의 일부가 될 다음 릴리즈 계획서 작성
- **팀 리더:**
  - 릴리즈 계획의 달성 가능 여부 확인
  - 릴리즈 번-업 차트 작성

4.4.4 릴리즈 계획 작업 산출물

릴리즈 계획에서는 다음의 작업 산출물이 작성된다.

- 인수 테스트가 수행될 시점을 포함해 릴리즈로 이어지는 각 이터레이션에서의 작업을 개략적으로 설명하는 릴리즈 계획서
- 팀 리더와 팀이 릴리즈를 위한 진행 상황을 식별하기 위해 사용하는 릴리즈 번-업 차트

Tip 14. 릴리즈 계획은 상위 레벨이다.

릴리즈 계획이 비전 또는 목표를 설명한다는 사실과 각 이터레이션의 세부 사항이 이터레이션 계획 미팅에서 결정될 것이라는 사실을 모든 사람이 알고 있어야 한다.

4.5 이터레이션 계획

이터레이션 계획

■ 목적

해당 이터레이션에서 구현할 사용자 스토리의 선택 및 이터레이션 진행 방법 계획

■ 설명

제품 책임자는 일반적으로 릴리즈 계획 및 사용자 스토리 우선 순위에 기반해 해당 이터레이션에서 구현하고자 하는 사용자 스토리를 선택한다. 팀은 선택된 사용자 스토리를 구현하고, 구현을 위해 요구되는 노력을 추정하는 데 필요한 모든 사용자 스토리를 확인한다.

제품 책임자와 팀 간의 협의를 통해 팀의 예상 속도에 맞는 이터레이션 백로그가 산출돼야 한다.

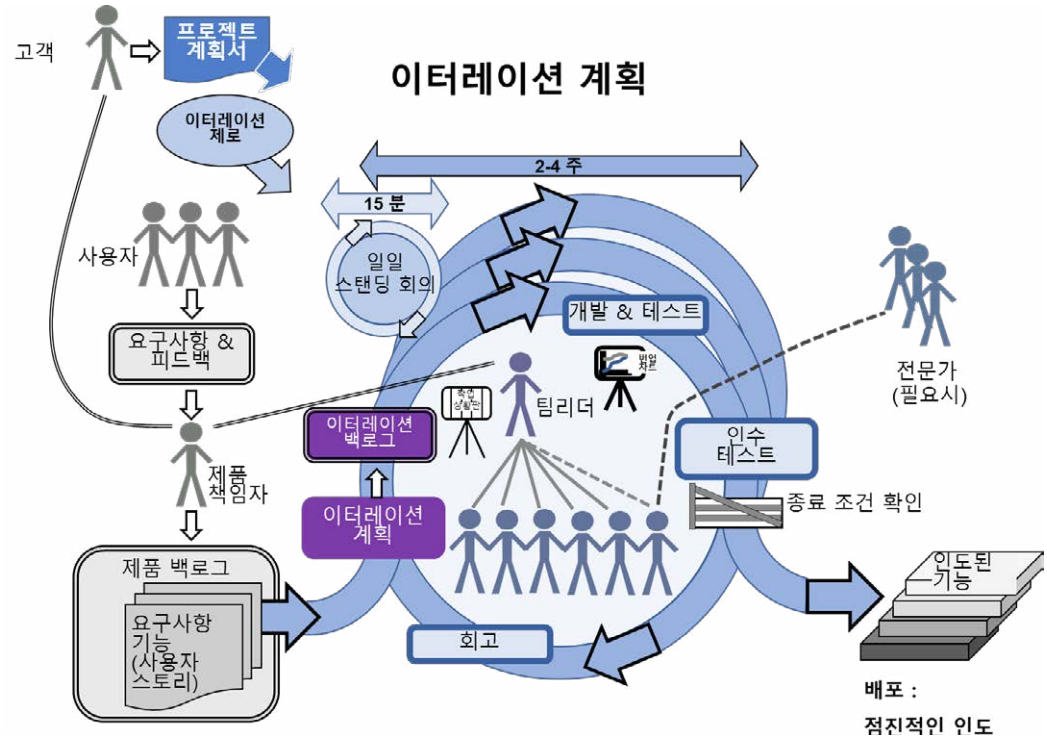
이터레이션 백로그의 내용이 합의된 후 팀은 스토리 인도 방법을 계획한다.

■ 작업 산출물

- 이터레이션 백로그 (사용자 스토리)–6.6
- 작업상황판–6.8

■ 역할

- 제품 책임자–3.2
- 팀 (팀 리더 및 팀원)–3.1



〈 그림 6. 구조도에서의 이터레이션 계획 〉

4.5.1 이터레이션 계획 시기

이터레이션 계획은 모든 이터레이션에서 첫 활동이다. 이것은 2개의 주요 부분 즉, 이터레이션 백로그 내용의 선택과 팀이 이터레이션 백로그를 인도할 방법에 대한 계획으로 구성된다.

4.5.2 이터레이션 계획이 무엇인가?

이터레이션 계획은 매 이터레이션 시작 시의 제한된 시간의(timeboxed) 두 번의 미팅으로 구성되는 데 이 미팅들은 서로 관련돼 있지만 참석자가 다소 변경된다.

이터레이션 백로그를 선택하는 첫 번째 미팅에서 제품 책임자와 팀은 이터레이션 동안 구현될 내용에 합의한다. 이 미팅의 참여자는 다음을 수행한다.

- 사용자 스토리가 필요한 항목의 식별
- 스토리들 사이의 의존성 식별, 이는 다른 스토리들도 필요함을 의미한다.
- 사용자 스토리를 완성하는 데 필요한 시간과 노력의 추정
- 자원의 가용성(예: 팀의 가용성, 속도 및 기술)을 기반으로 해당 이터레이션 내에서 완성될 수 있는 사용자 스토리의 선택

팀의 활동을 계획하는 두 번째 미팅에서 팀은 이터레이션 백로그에 포함시키기로 합의된 스토리의 인도 방법을 결정한다. 이 미팅의 참여자는 다음을 수행한다.

- 누가 어떤 작업(예: 설계, 프로그래밍, 스토리 테스팅, 인수 테스팅, 배포)을 담당할 것인지와 이터레이션 백로그에서 누가 어떤 스토리를 담당할 것인지의 결정
- 스토리를 인도하는 데 어떤 전문적인 도움이 필요한지 식별
- 스토리들 사이의 의존성을 기반으로 이터레이션 기간 동안의 작업 스케줄 결정
- 관련 사용자를 인수 테스팅에 참석시키기

4.5.3 이터레이션 계획 역할

이터레이션 계획 참여자의 역할은 다음과 같다.

■ 제품 책임자:

- 일반적으로 릴리즈 계획 및 사용자 스토리 우선 순위를 기반으로 이터레이션에서 구현하려는 사용자 스토리를 선택한다.
- 팀과 협의해 선택된 사용자 스토리들 (및 이 스토리들이 의존하는 다른 사용자 스토리들) 중의 어떤 것이 해당 이터레이션에서 인도될 수 있을지를 확정한다.

■ 팀 리더:

- 팀 및 제품 책임자와 협의해 달성 가능한 이터레이션 백로그에 대해 합의한다.
- 이터레이션 백로그에 있는 스토리를 구현하기 위한 작업 스케줄을 계획한다.
- 팀원에게 작업을 할당한다.
- 요구되는 모든 전문적 도움이나 자원에 대해 합의하고 준비한다.

■ 팀원:

- 선택된 사용자 스토리의 구현을 지원하는 데 필요한 다른 사용자 스토리들을 식별한다.
- 각 사용자 스토리를 구현하는 데 필요한 노력을 추정한다.
- 해당 이터레이션에서 수행할 작업을 계획하는 일에 도움을 준다.
- 요구되는 모든 전문적 도움이나 자원을 식별한다.

4.5.4 이터레이션 계획 작업 산출물

이터레이션 계획 참여자들은 다음을 작성하고 업데이트한다.

- 이터레이션 백로그
- 작업상황판

이러한 작업 산출물을 작성하기 위해 이터레이션 계획에는 다음이 포함된다.

- 제품 책임자 및 모든 팀이 참석하는 제한된 시간의(timeboxed) 이터레이션 계획 미팅
- 아래를 기반으로 하는 이터레이션 팀의 예상 속도 계산
  - 팀의 평균 속도 (팀이 일반적인 이터레이션에서 인도하는 작업량)
  - (휴일, 질병, 관리 작업, 교육 등의 이유로 예상되는 결근을 참작한) 해당 이터레이션을 위한 팀원의 가용성
- 사용자 스토리들의 우선 순위 지정
- 해당 사용자 스토리를 지원하는 데 필요한 다른 사용자 스토리들 및 기술적 스토리들에 대한 의존성 식별
- 해당 이터레이션에 필요한 추가적인 전문적 도움 또는 자원의 식별
- 사용자 스토리의 인도에 필요한 예상 노력의 추정
- 추정된 노력이 팀의 예상 속도와 일치하는지 평가
- 이터레이션 백로그를 해당 이터레이션에 맞추기 위한 스토리의 추가 또는 축소
- 의존성에 유의하면서 이터레이션 백로그의 스토리들을 구현하는 데 필요한 작업 계획
- 해당 이터레이션에 수행될 작업의 할당에 대한 팀원의 합의
- 합의된 모든 사용자 스토리 및 기술적 스토리로 작업상황판 구성

Tip 15. 제품 책임자는 항상 보다 많은 것을 원한다!

고객을 대표하는 역할 때문에 제품 책임자가 기능이 최대한 빨리 사용자에게 인도되기를 원하는 것은 당연한 일이다. 이것은 제품 책임자가 매 이터레이션마다 보다 많은 사용자 스토리를 인도하도록 팀에 요구하는 것을 의미한다. 효과적이고 효율적인 운영을 위해 팀이 지속 가능한 페이스(sustainable pace)를 유지할 필요가 있다. 그렇지 않으면 팀이 '탈진(burn-out)'하게 돼 모두가 고통 받을 것이다. 대부분의 팀은 제품 책임자로부터 보다 많은 작업에 대한 요구를 수용하려 할 것이지만 팀 리더는 팀을 보호하면서 인도할 수 있는 것보다 많은 것을 팀이 약속하지 않도록 해야 한다. 모든 사람이 팀의 속도(번-업 차트에 나타나는 이터레이션당 작업 산출량)를 알 수 있어야 하며 이것이 초과되기를 기대하는 것은 단기적으로는 이익을 가져올지 몰라도 생산성(또는 팀원)에 있어서 장기적인 손실을 초래할 것이다.

Tip 16. 이터레이션 백로그 변경 사항

이터레이션 백로그는 일단 합의되면 이터레이션 동안 변경돼서는 안 된다. 제품 책임자는 이터레이션이 진행되는 동안 사용자 스토리를 추가하거나 삭제할 수 있다고 생각해서는 안 된다.

4.6 일일 스탠드 업 미팅

일일 스탠드 업 미팅

■ 목적

모든 팀원이 현재의 진행 상황을 알 수 있게 하는 것

■ 설명

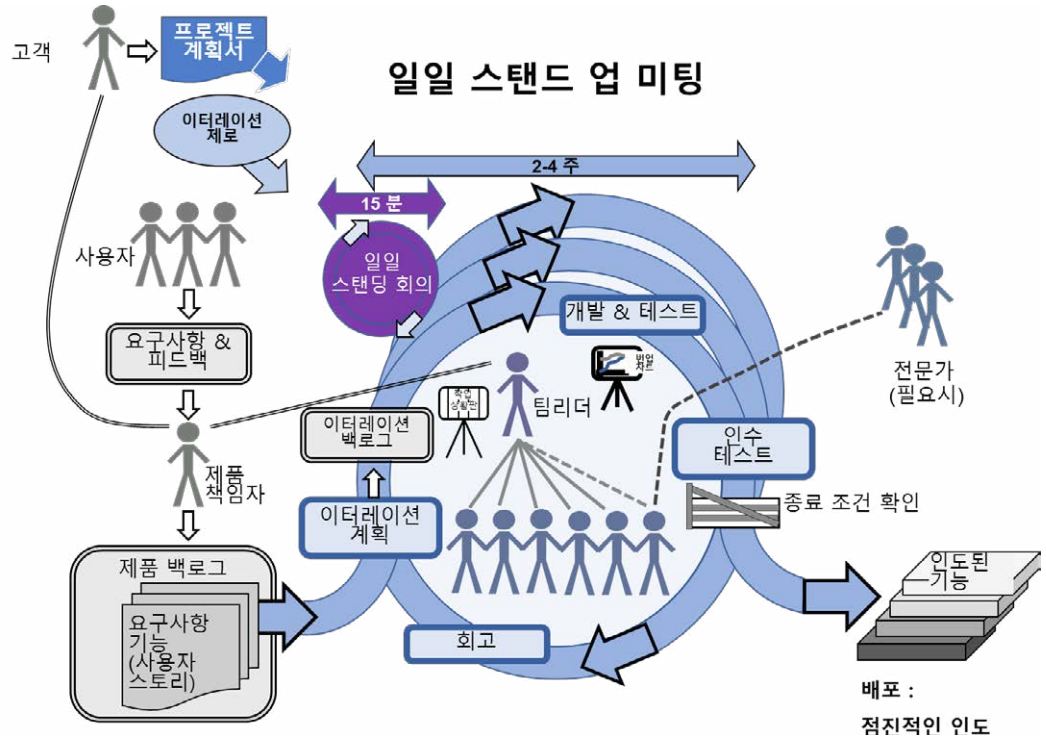
매일 15분으로 제한된 시간(timeboxed) 동안 모든 팀원이 이 미팅에 참석한다. 각 팀원은 팀 리더의 도움을 받아 지난 미팅 이후에 달성한 것, 다음 미팅 전에 달성하고자 하는 것 및 작업 진행에 방해가 되는 모든 것을 진술한다. 미팅은 팀의 작업 진행에 도움이 되도록 참가자들 사이의 후속 논의를 촉구한다. 팀 리더는 보고되는 진행 상황과 작업상황판이 일치하는지 확인한다.

■ 작업 산출물

- 작업상황판–6.8

■ 역할

- 팀 (팀 리더 및 팀원)–3.1



〈 그림 7. 구조도에서의 일일 스탠드업 미팅 〉

4.6.1 일일 스탠드업 미팅 시기

일일 스탠드업 미팅은 당연히 매일 같은 장소에서 진행된다. 근무 일의 시작 무렵 모든 팀원이 참석할 수 있는 시점에 미팅을 갖는 것이 좋다.

4.6.2 일일 스탠드업 미팅이 무엇인가?

일일 스탠드업 미팅은 팀의 진행 상황 미팅이다. 이것은 짧지만 매우 필수적이다. 미팅의 목적은 모든 팀원이 진행 상황, 방해 사항 및 당일의 계획을 알 수 있도록 하는 것이다.
일일 스탠드업 미팅에 이어서, 참석자들 (필요한 경우, 제품 책임자, 사용자 및 기타 전문가 포함) 사이의 추가적인 논의가 필요할 수 있다.

4.6.3 일일 스탠드업 미팅 역할

일일 스탠드업 미팅의 참석자는 다음과 같다.

- **팀 리더**
  - 팀원을 돕고 제한 시간을 지킨다.
  - 미팅에 기반해 작업상황판을 업데이트한다.
  - 해결할 필요가 있는 방해 사항들을 기록한다 (후에 조치를 취한다).
- **팀원 (또는 팀과 공동 작업하는 전문가)**
  - 작업 진행 상황과 완성된 작업에 대해 간략하게 보고한다.
  - 다음에 수행할 작업에 대해 간략하게 보고한다.
  - 도움이 요구되는 진행 상의 방해 사항에 대해 간략하게 보고한다.
  - 다른 팀원의 보고를 경청한다.
- **옵서버(observer)로서 제품 책임자**
  - 미팅을 지켜보면서 경청한다.

4.6.4 일일 스탠드업 미팅 작업 산출물

일일 스탠드업 미팅에서의 작업 산출물은 업데이트된 작업상황판이다.

팀 리더는 일일 스탠드업 미팅에서 보고되는 정보가 올바르게 작업상황판에 반영돼 있는지 확인하고 필요한 경우 업데이트한다. 이상적으로는 작업을 완료한 팀원이 이미 그 작업을 작업상황판에 업데이트했어야 한다.

Tip 17. 일일 스탠드업 미팅 – 진행 시간을 짧게 하기

일일 스탠드업 미팅에서는 진행 상황과 계획이 신속하게 파악돼야 한다. 미팅을 15분 내에 끝내려면 자제와 요약이 필요하다. 방해 사항을 해결하고 싶더라도 그것은 미팅 후에 처리돼야 한다. 그렇지 않으면 자신들에게 무의미한 논의를 경청하는 일에 다른 팀원의 시간이 낭비될 수 있다.

4.7 개 발

개 발

■ 목적

사용자 스토리를 코드로 구현하는 것

■ 설명

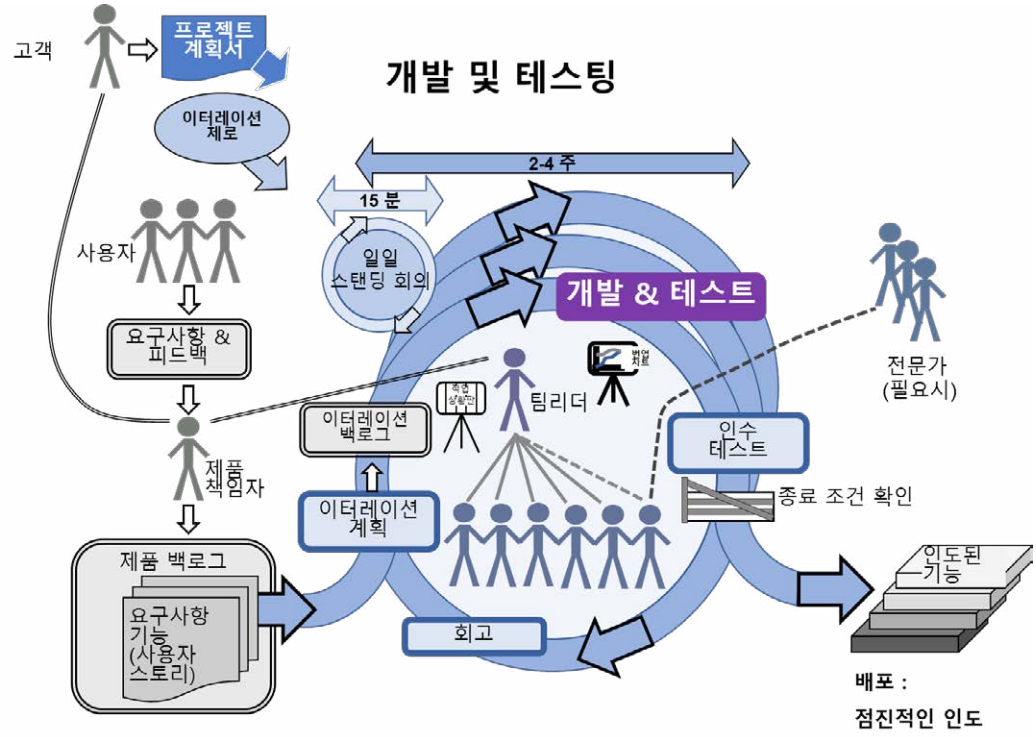
이터레이션 백로그에 있는 사용자 스토리를 설계하고 코드화한다. 개발자는 사용자 스토리와 관련된 요구 사항을 완전히 파악하고 팀 리더와의 합의로 문서화된 관련 설계를 작성한다.
사용자 스토리는 이 설계에 기반해 코드화되고 단위 테스트가 작성되며 정적 분석(static analysis)을 통과한 코드는 빌드(build)에 포함된다.

■ 작업 산출물

- 문서화된 설계
- 코드
- 자동화된 단위 테스트
- 소프트웨어 빌드

■ 역할

- 팀-3.1



〈 그림 8. 구조도에서의 개발 〉

4.7.1 개발 시기

개발은 이터레이션 계획 이후와 인수 테스팅 전에 각 이터레이션의 전체 기간 동안 진행된다.

4.7.2 개발은 무엇인가?

개발은 프로젝트의 핵심이다. 이것의 목적은 사용자 스토리를 코드로 구현하는 것이다. 팀은 사용자에게 유용한 동작하는 소프트웨어를 설계하고 개발한다. 개발에는 해당 소프트웨어에 대한 단위 테스팅도 포함된다.

4.7.3 개발 역할

개발 참여자는 다음과 같다.

■ **팀 리더**

- 설계를 주도한다.
- 다른 개발자들을 코치한다.

■ **팀 리더로부터 다음을 수행하도록 요구 받은 팀원 또는 전문가**

- 기술적 설계를 주도한다.
- UX 설계를 주도한다.
- 다른 개발자들을 코치한다.

■ **팀 리더, 팀원 또는 전문가**

- 소프트웨어를 설계하고 코드화하며 단위 테스트를 수행한다.
- 다른 개발자들의 설계 및 코드를 리뷰 한다.
- 결함(defects) 해결을 통해 스토리 테스팅과 인수 테스팅을 지원한다.

4.7.4 개발 작업 산출물

개발 작업 산출물은 다음과 같다.

- 문서화된 설계 – 기술적 설계 및 UX 설계가 필요할 수 있다.
- 자동화된 단위 테스트로 테스트된 코드
- 소프트웨어 빌드(build)

이러한 산출물들을 작성하는 개발자의 작업에는 다음이 포함된다.

- 제품 책임자 및 관련 사용자와 상의해 사용자 스토리에 필요한 것을 완전히 파악
- 사용자 스토리를 위한 설계 및 리뷰
- 사용자 스토리에 대한 단위 테스트의 설계 및 자동화
- 코드 작성, 코드에 대한 단위 테스팅 및 디버깅, 코드 및 테스트 리뷰 및 정적 분석(static analysis) 수행
- 필요에 따라 다른 개발자들의 설계 및 코드 리뷰

코드가 빌드에 포함되려면 단위 테스트, 리뷰 및 정적 분석을 통과했음이 확인돼야 한다.

Tip 18. 회고(Retrospectives)

팀마다 제각기 다른 개발 프랙티스를 가지고 있다. 해당 팀이 많은 이터레이션(iterations)을 통해 리드미를 경험하는 동안 그 팀의 필요에 맞추어 회고(Retrospective)를 개발 프랙티스 개선의 기회로 활용할 수 있다.

항상 지속 가능한 페이스(sustainable pace)로 작업하라.

Tip 19. 지속 가능한 페이스(sustainable pace)

팀은 무기한 지속될 수 있는 페이스(pace)로 작업하므로, 최적의 작업을 방해해 개발자와 테스터를 탈진시킬(burn out) 수 있는 빈번한 초과 작업 시간과 장시간 작업을 피할 수 있도록 작업량(workloads)이 관리되야 한다.

리드미를 채택할 때 사용할 수 있도록 몇 가지의 첫 단계 프랙티스를 권장한다. 참조사항에는 추가적으로 고려할 만한 몇 가지 선택적인 고급 접근법이 있다.

Tip 20. 권장되는 첫 단계 개발 프랙티스

간결한 설계: 미래에 유용할 수 있는 것이 아니라 현재 필요한 것에 기반한 설계; 하지만 이 접근법은 실용적이어야 하며 한 두 번의 이터레이션만 수행할 경우에는 적용되지 않는다. 어떤 기능이 다음 몇 번의 이터레이션에 추가될 것이라면 해당 설계에 그 기능이 고려돼야 한다. 설계하는 동안 현재 이터레이션에 일어나고 있는 일만 보길 바란다.

기능 주도 개발방법론(Feature-Driven Development, FDD): 팀은 아키텍처 설계보다는 기능을 먼저 개발한다. 따라서 기능이 이미 완전히 통합돼 있으므로 통합 단계가 필요하지 않다.
코드 우선(code-first) 프로그래밍: 코드 우선 접근법은 코드를 먼저 개발하고 단위 테스트를 수행해 코드가 올바르게 작동하는지 확인한다.

지속적 통합(Continuous Integration, CI): 하나의 코드가 공유 저장소(shared repository)에 저장될 때마다 자동화된 회귀 테스트가 수행돼 새로운 코드가 기존 코드에 어떤 영향도 미치지 않았음을 확인한다.
지속적 통합에는 빌드 자동화 및 테스트(automated Build & Test)가 포함되는데 이것은 소스 코드로부터 빌드를 자동화하고 그 빌드가 공유 저장소에 저장될 때 자동적으로 테스트한다.

일단 위의 접근법들이 적용되면 보다 진보된 다른 접근법들도 고려할 수 있다.

4.8 스토리 테스팅

스토리 테스팅

■ 목적

사용자 스토리 구현의 검증

■ 설명

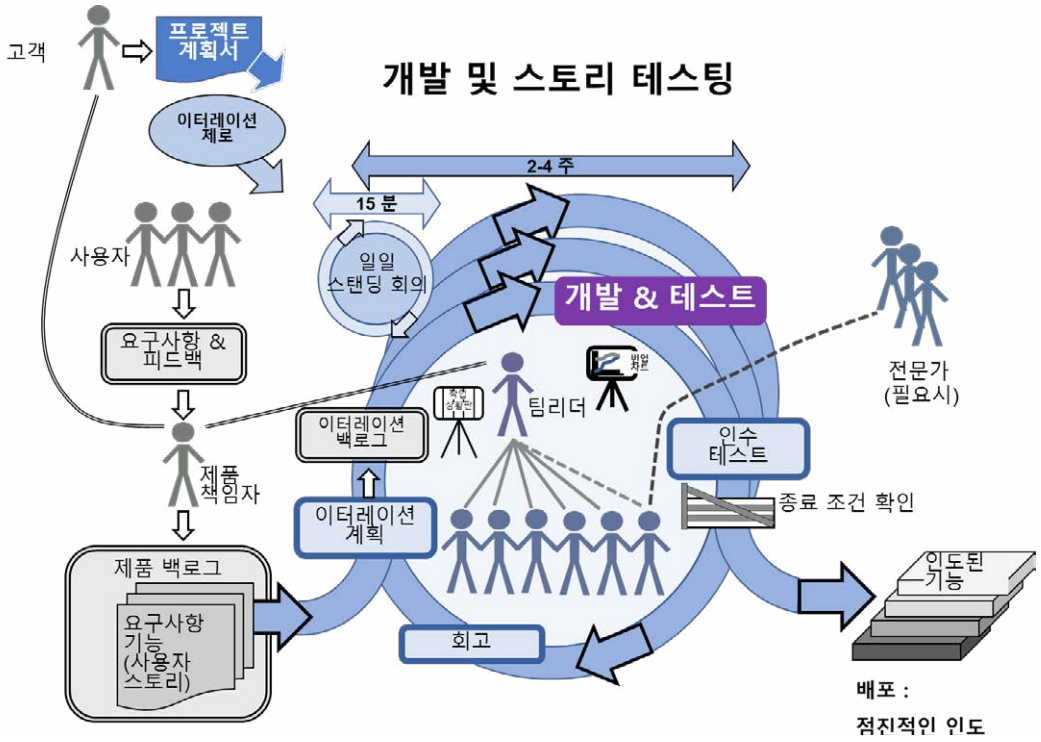
이터레이션 백로그에 있는 사용자 스토리에 대해 테스트한다.
테스터는 관련 이해 관계자들과의 대화를 통해 사용자 스토리와 관련된 요구 사항을 완벽하게 파악해야 한다. 다음으로 리스크에 기반한 접근법을 활용해 관련된 (기능적 및 비기능적) 테스트를 수행한다.
탐색적 테스팅(Exploratory Testing)은 초기에 사용되지만 가장 높은 리스크의 테스트케이스는 테스트를 자동화해 회귀 테스팅이 자동화되도록 한다.

■ 작업 산출물

- 스토리 테스트 결과물
- 자동화된 회귀 테스트

■ 역할

- 팀-3.1



〈 그림 9. 구조도에서의 스토리 테스팅 〉

4.8.1 스토리 테스팅 시기

스토리 테스팅은 사용자 스토리를 위한 개발에 따라 매 이터레이션 전체에 걸쳐서 진행된다. 스토리 테스팅을 통과하는 코드는 인수 테스팅을 위한 준비가 된 것이다.

4.8.2 스토리 테스팅이 무엇인가?

스토리 테스팅은 개발된 코드가 사용자가 요구하는 기능임을 확인하기 위해 팀이 수행한다. 이를 검증(validation)이라고 하며 사용자의 필요를 충족시키는 사용자 스토리가 인도되었는지 확인한다. 또한 이전에 인도된 코드가 새로운 사용자 스토리에 의해 동작하는지 확인하기 위한 회귀 테스팅도 포함된다.
스토리 테스팅은 이터레이션 백로그의 각 사용자 스토리에 대해서 수행된다.

4.8.3 스토리 테스팅 역할

스토리 테스팅은 팀이 수행한다. 스토리 테스팅의 참여자들은 팀원, 팀 리더 또는 전문적 테스터일 수 있다. 스토리 테스터는 다음을 수행한다.

- 스토리 테스트를 설계하고 수행한다.
- 스토리 테스트 및 그 결과를 검토한다.
- 스토리 테스트를 자동화해 회귀 테스트 수트(suite)의 일부로 편입시킨다.
- 스토리 레벨 회귀 테스트를 수행한다.
- 전문적 테스팅(예: 성능, 보안, 사용성, 신뢰성)을 주도한다.
- 다른 테스터들을 코치한다.

4.8.4 스토리 테스팅 작업 산출물

스토리 테스팅 작업 산출물은 다음과 같다.

- 스토리 테스트 결과물
- 자동화된 회귀 테스트

스토리 테스터는 이러한 산출물의 작성에 기여하면서 다음의 작업을 수행한다.

- 제품 책임자 및 사용자와 대화해 요구 사항이 완전히 파악되었는지 확인한다.
- 다음과 같은 방법으로 사용자 스토리 및 기술적 스토리에 대한 기능적 스토리 테스트와 비기능적 스토리 테스트를 설계해 작성한다.
  - 각 스토리와 관련된 리스크에 중점을 두면서 얼마나 많은 테스팅을 수행할지 결정한다.
  - 스크립트된(scripted) 테스팅 접근법과 탐색적 테스팅 접근법을 모두 사용한다.
- 전체 회귀 테스트 수트(suite)에 추가될 수 있는 회귀 테스트를 작성하기 위해 최상위 리스크 테스트를 자동화한다.
- 테스트 설계, 테스트케이스 및 테스트 결과를 검토한다.

Tip 21. 프로젝트 요구 사항에 맞도록 테스팅 프랙티스 개선하기

팀들마다 제각기 다른 테스팅 프랙티스를 가지고 있다. 해당 팀이 많은 이터레이션(iterations)을 통해 리드미를 경험하는 동안 회고(Retrospective)를 테스팅 프랙티스 개선의 기회로 활용할 수 있다.

리드미를 채택할 때 사용할 수 있도록 몇 가지의 첫 단계 프랙티스를 권장한다. 참조사항에는 추가적으로 고려 할 만한 몇 가지 선택적인 고급 접근법이 있다.

Tip 22. 권장되는 첫 단계 스토리 테스팅 프랙티스

통합 테스팅(Integrated testing): 이상적인 이터레이션이라면 모든 테스팅은 공동 작업하는 팀에 의해 해당 이터레이션 내에 수행된다. 이것은 개발자와 테스터가 공동 작업하고 서로를 동등한 지위로 대우하며 인도 가능한 산출물의 품질에 대해 공동으로 책임지는 통합 접근법(integrated approach)이다.

탐색적 테스팅(Exploratory Testing): 학습, 테스트 설계 및 수행이 동시에 이루어진다. 각 테스트 결과가 탐색돼야 할 다음 테스트 영역에 방향을 제시해 준다. 탐색적 테스팅 중에 노트 기록(note-taking)을 철저히 함으로써 테스트 과정과 결과로부터 추가적인 테스팅을 위한 영역 및 잠재적 결함을 알 수 있다.

스토리 회귀 테스팅(Story Regression Testing): 해당 이터레이션에서의 변경이 이전에 완료된 작업에 나쁜 영향을 주지 않음을 확인하기 위한 테스팅이다. 변경은 소프트웨어나 인프라에 대한 변경일 수 있다. 회귀 테스트 대상은 매우 빠르게 확장되므로 리드미는 사용자 스토리에 대한 회귀 테스팅의 작업량을 제어할 수 있는 몇 가지 방법을 권장한다.

- 높은 리스크의 사용자 스토리 테스트를 자동화해 모든 이터레이션에서 자동화를 수행하라.
- 단위 테스트의 회귀 테스트에 스토리 회귀 테스트를 포함시킬 것을 고려하라.
- 스토리 회귀 테스트들을 목록화하고 이터레이션에서 리스크에 따라 수행될 부분을 선택하라.

비공식 결함관리(Informal Defect Management): 스토리 테스팅 중에 이터레이션 동안 수정돼야 할 결함이 발견될 것이다. 어떤 이슈가 현재의 이터레이션에서 수정될 수 있는 경우 그것을 공식적으로 문서화할 필요는 없다. 현재의 이터레이션에서 수정될 수 없는 경우 제품 백로그에 그것을 추가하라.

일단 이러한 접근법들이 적용되면 보다 진보된 다른 접근법들도 고려할 수 있다.

4.8.5 스토리 테스팅의 유형

스토리 테스팅은 여러 가지 방법으로 접근할 수 있다.

- 탐색적 접근법으로 수행한다.
- 스크립트된(scripted) 테스트로 수행한다.
- 일반적으로 탐색적 테스트와 스크립트된 테스트를 조합할 경우 가장 잘 수행된다.

다음 사항이 권장된다.

- 스토리 테스팅에 대한 탐색적 접근법으로 시작할 것
- 회귀 테스팅 및 자동화를 위한 스토리 테스트를 작성하기 위한 테스트 설계 기법을 사용할 것

스토리 테스트는 팀에 의해 작성되며 다음의 테스트를 포함한다.

- 인수 기준
- 용자 오류 및 실수
- 기술적 리스크

탐색적 접근법이 사용되고 있는 경우에는 테스트 차터가 작성될 수 있으며 명세화된 테스트 케이스나 자동화된 테스트 스트립트가 있을 수 있다.

이러한 접근법에 대한 자세한 내용은 5.8 테스팅을 참고하라.

4.8.6 전문적 테스팅

전문적 테스팅은 스토리 테스팅의 한 구체적 형태이다. 다음의 기간에 수행될 수 있다.

- 이터레션 동안
- 비정기적인 특별 이터레이션 동안
- 병행 이터레이션(parallel iteration) 동안

전문적 테스팅은 프로젝트 기간 동안에 필요한 모든 시점에 수행된다.
전문적 테스팅은 일반적으로 소프트웨어의 비기능적 요구 사항에 대해 수행되며 다음과 같다.

- 성능 테스팅
- 안 테스팅
- 용성 테스팅
- 접근성 테스팅
- 신뢰성 테스팅

어떤 프로젝트에서는 이러한 테스트들이 덜 중요하며 스토리 테스팅의 일환으로 수행되거나 전혀 수행되지 않을 수 있다. 어떤 프로젝트에서는 그 중 하나 이상이 필요할 수 있다.
전문적 테스팅 역할은 스토리 테스팅과 동일하지만 전문적 영역에 중점을 둔다.
전문적 테스팅 작업 산출물도 스토리 테스팅과 동일하지만 전문적 영역에 중점을 둔다.

Tip 23. 전문적 테스트에는 추가적인 도구가 필요할 수 있다.

전문적 테스트는 규모가 매우 클 수 있고 전문적 테스터와 전문적 도구가 필요할 수 있으며 특수한 환경에서 수행돼야 할 수도 있다.

예제 14. 팀 리더가 기술적 설계의 리더인 경우

▶ 'PIP' 프로젝트에는 성능 및 보안 테스팅을 위한 특수한 환경이 요구된다.

'무영'은 'PIP' 프로젝트의 제품 책임자이고 '훈'은 팀 리더다. 사용자가 성능 및 보안 인수 기준의 확인을 위해 자신을 대신해서 성능 및 보안 전문가가 "실제 같은" 성능 및 보안 테스트를 수행할 수 있는지 문의하였기 때문에 이들은 전문적 스토리 테스트를 준비해야 한다. '무영'과 '훈'은 이 테스트가 보통의 이터레이션 스토리 테스트 또는 인수 테스트 내에서는 불가능할 것임을 인식하고 있다. '훈'은 "보안 및 성능 테스팅을 위해 전문적 테스터에게 부탁해야 할 것이며 실제 같은 환경을 예약해야 할 것이다. 이 환경은 6주 후에 이용 가능하다."라고 말한다. '무영'은 "알겠다. 6주 후에 특별 테스트 이터레이션을 가지는 것에 대해 사용자와 대화하겠다."라고 대답한다.

4.9 인수 테스팅

인수 테스팅

■ 목적

사용자 및 제품 책임자의 사용자 스토리에 대한 구현의 확인 및 릴리즈

■ 설명

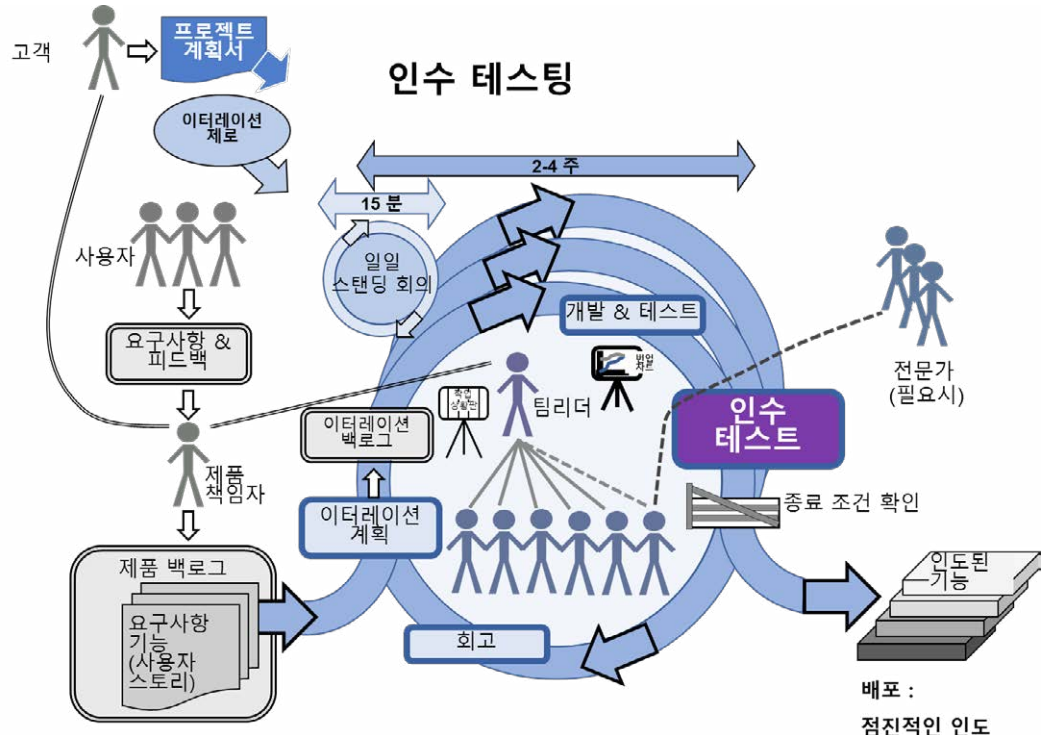
인수 기준에 기반한 인수 테스트가 팀에 의해 개발돼 인수 테스팅을 수행하도록 사용자에게 제공된다. 사용자는 피드백을 제공한다. 제품 책임자는 어떤 코드가 릴리즈를 위해 준비돼 있는지에 대한 최종 의견을 낸다.

■ 작업 산출물

- 문서화된 인수 테스트 수트
- 인수 테스트 결과물
- 인수될 코드

■ 작업 산출물

- 팀-3.1
- 사용자-3.6
- 제품 책임자-3.2



〈 그림 10. 구조도에서의 인수 테스팅 〉

4.9.1 인수 테스팅 시기

인수 테스팅은 해당 이터레이션에서 개발된 모든 기능에 대해 일괄적으로 수행된다. 이 테스팅은 스토리 테스팅 후와 배포 전에 수행된다.

4.9.2 인수 테스팅이 무엇인가?

제품 책임자 및 사용자가 이터레이션 동안 구현된 각 사용자 스토리를 테스트하는 것으로 해당 소프트웨어가 요구 사항을 충족해 "릴리즈"될 지의 여부를 결정할 수 있게 한다. 일반적으로 각 사용자 스토리는 별개의 것으로 간주되므로 해당 이터레이션에 구현된 어떤 사용자 스토리는 릴리즈를 위해 인수될 수 있지만 어떤 것은 인수되지 못한다.

인수 테스팅의 일환으로 인도될 기능이 인수 기준(사용자 스토리의 일부), 완료 정의 및 (상황이 변했을 수 있으므로) 사용자가 고려하기 원하는 그 밖의 다른 것과 비교된다. 사용자는 해당 이터레이션과 인도된 것에 대해 피드백을 제공한다.

제품 책임자는 어떤 코드가 릴리즈를 위해 준비돼 있는지에 대한 최종 의견을 내린다.

4.9.3 인수 테스팅 역할

인수 테스팅은 팀의 도움을 받아 사용자와 제품 책임자가 수행한다. 인수 테스팅의 참여자들은 다음과 같다.

■ **사용자**

- 팀이 제공하는 인수 테스트 세트 수행
- 해당 소프트웨어를 탐색해 요구사항에 맞게 작동하는지 확인
- 해당 소프트웨어 및 프로세스의 이슈 사항 보고

■ **제품 책임자**

- 인수 테스트에 옵서버(observer)로 참여
- 인수 테스트의 수행 및 결과 보고
- 사용자 및 팀과 함께 해결해야 할 결함의 식별
- 해당 이터레이션에서 해결되지 않은 결함을 제품 백로그에 추가
- 릴리즈 준비가 된 소프트웨어와 그렇지 않은 소프트웨어의 확정

■ **팀원 또는 팀 리더**

- 사용자 스토리를 기반으로 한 인수 테스트 세트의 설계
- 인수 테스트를 회귀 테스트 수트(suite)의 일부로 자동화
- 보고된 결함(defects)를 해결함으로써 인수 테스팅을 지원
- 테스트 환경을 설정하고 사용자의 테스트 수행 돕기

4.9.4 인수 테스팅 작업 산출물

인수 테스팅 작업 산출물은 다음과 같다.

- 문서화된 인수 테스트 수트
- 인수 테스트 결과물
- 인수될 코드

Tip 24. 팀들마다 제각기 다른 인수 테스팅 프랙티스가 있다.

해당 팀이 많은 이터레이션(iterations)을 통해 리드미를 경험하는 동안 회고(Retrospective)를 인수 테스팅 개선의 기회로 활용할 수 있다.

리드미를 채택할 때 사용할 수 있도록 몇 가지의 첫 단계 프랙티스를 권장한다. 참조사항에는 추가적으로 고려할 만한 몇 가지 선택적인 고급 접근법이 있다.

Tip 25. 권장되는 첫 인수 테스팅 프랙티스

비공식 결함관리: 인수 테스팅 중에 이터레이션 동안 해결돼야 할 결함이 발견될 것이다. 어떤 이슈가 현재의 이터레이션에서 수정될 수 있는 경우 그것을 공식적으로 문서화할 필요는 없다. 현재의 이터레이션에서 수정될 수 없는 경우 제품 백로그에 그것을 추가하라.
중요한 높은 리스크의 인수 테스트를 자동화해 미래의 이터레이션을 위한 스토리 테스팅 회귀 세트의 일부가 되게 하는 일은 유용하다. 일단 이러한 접근법들이 적용되면 보다 진보된 다른 접근법들도 고려할 수 있다.

예제 15. 너무 바빠서 인수 테스트를 할 수 없는 사용자

▶ 너무 바빠서 인수 테스트를 할 수 없는 사용자

'여진'은 시간이 없다. '여진'은 '마이트래블(My-Travel)' 프로젝트 사용자 중 한 명이다. 그녀는 '마이트래블' 팀 리더인 '요한'과 '마이트래블' 제품 책임자인 '숙자'를 만난다. '여진'은 지난 이터레이션에 사용자 스토리 인수 테스트를 했고 그것에 많은 시간을 썼다. 지금 팀은 그녀에게 해당 이터레이션(iteration)에 대한 인수 기준 및 인수 테스트 세트를 확인해 줄 것을 요구했다. '여진'이 소리친다. "나는 이것을 할 시간이 없어요! 팀이 나에게 도움을 청한 지 이제 겨우 3주 밖에 안 되었는데 테스트 세트 정의와 다음 주에 인수 테스팅을 도와달라고 하는군요. 나는 마무리해야 할 중요한 과제가 있어요. 해야 할 일이 있다구요! 당신 팀을 도와줄 수가 없어요!"

'숙자'는 잠시 생각한 후 말한다. "음, 어쩌면 도움이 필요한 횟수를 줄일 수 있겠어요. '요한', 매 이터레이션마다 소프트웨어를 릴리즈하는 대신, 매 4번째 이터레이션(iteration)마다 인수 테스트를 수행한 후 릴리즈하는 건 어때요? 가능한가요?" '요한'이 동의한다. "네, 다음 이터레이션부터 그렇게 할게요. 이번 이터레이션에는 '여진'의 사용자 스토리를 축소해도 될까요?" '숙자'는 고개를 끄덕이며 묻는다. " 어떻게 생각해요, '여진'? 이번 이터레이션에서 스토리를 빼도 기다릴 수 있겠어요? 이번 이터레이션에 스토리를 원한다면 테스트를 수행해야 해요!"

'여진'이 웃는다. " 옳아요. 내가 스토리를 인수하지 않는다면 테스트하지 않아도 되는군요. 좋아요. 매 4번째 이터레이션에 인수 테스팅하는 아이디어가 마음에 들어요. 이번 이터레이션에 내 스토리를 축소하는 것을 수락할게요." '요한'이 동의한다. " 스토리는 스토리 테스트 종료 무렵에 끝내도록 할게요. 프로세스 변경에 대해서는 팀과 얘기할 게요." '숙자'가 말한다. "좋아요. 다른 사용자와 얘기해서 이 변경이 그들에게 어떻게 영향을 미치는지 알아보겠어요. 대부분의 사람들이 괜찮다면, 다음 이터레이션부터 매 4주마다 인수 테스팅 및 배포와 함께 릴리즈 계획을 수행하도록 변경하겠어요."

4.9.5 인수 테스팅의 유형

인수 테스팅에는 여러 가지 접근법이 있으며 다음 중 하나 이상이 포함된다.

- 자신이 필요로 하는 모든 것이 작동되는 지를 확인하기 위해 사용자 스토리를 여러 모로 체험하고 자신의 작업을 통해 수행하는 제품 사용자
- 인수 기준에 대해 구체적으로 합의된 테스트의 수행
- 소프트웨어에 관해 질문하고 새로운 기능을 사용자에게 설명하기
- 제 3자(다른 사람 또는 조직)에게 사용자를 대신해 인수할 것을 요구하기

팀은 사용자에게 인수 테스트 세트를 제공한다. 이것은 해당 소프트웨어가 인수 기준을 충족시키는 지의 여부를 테스트하는 방식으로 사용자 스토리를 실행하기 위한 지침이다.

이러한 인수 테스트 세트에는 탐색적 접근법 또는 스크립트된 접근법이 사용될 수 있다.

이러한 접근법에 대한 자세한 내용은 **5.8 테스팅**을 참고하라.

4.10 배포

배 포

■ 목적

동작하는 소프트웨어를 설치해 사용될 수 있도록 준비하기

■ 설명

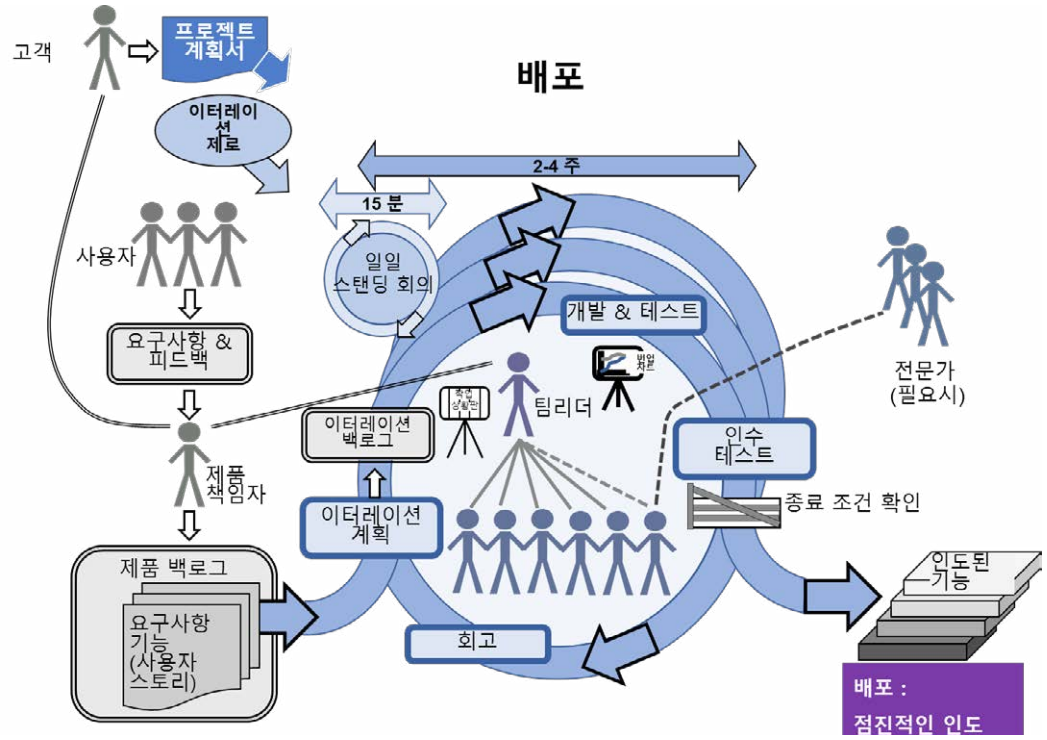
배포 동안 팀은 소프트웨어를 생산 환경에 설치하고 확인한다. 다른 팀들 및 IT 인프라 팀들과 연락을 취한다. 사용자와 함께 작업해 사용자가 새 배포물에 익숙해지도록 한다.

■ 작업 산출물

- 배포 메뉴
- 배포되는 실행 파일

■ 역할

- 팀-3.1
- 필요한 경우 다른 팀
- 필요한 경우 IT 인프라 팀



〈 그림 11. 구조도에서의 배포 〉

4.10.1 배포(Deployment)가 무엇인가?

배포 동안에 팀은 생산 환경에 소프트웨어를 설치해 사용자가 자신의 작업을 수행할 수 있게 한다. 배포는 자동화를 사용해 수동적으로 또는 지속적으로 수행될 수 있다.

4.10.2 배포 목적 및 시기

배포의 목적은 생산 환경에서 동작하는 소프트웨어가 사용될 수 있게 하는 것이다.

4.10.3 배포 역할

팀원은 소프트웨어를 배포하고 그것이 올바르게 배포되었는지 확인할 책임이 있다.

4.10.4 배포 작업 산출물

배포 작업 산출물은 배포되는 소프트웨어와, 배포되는 대상, 배포 동안 수행될 테스트 방법 및 필요한 모든 승인을 설명하는 배포 메뉴이다.

배포에 대한 자세한 내용은 **5.10.2 지속적 배포**를 참고하라.

4.11 회고 미팅

회고 미팅

■ 목적

개발 프로세스에서 개선될 사항을 식별할 기회 갖기

■ 설명

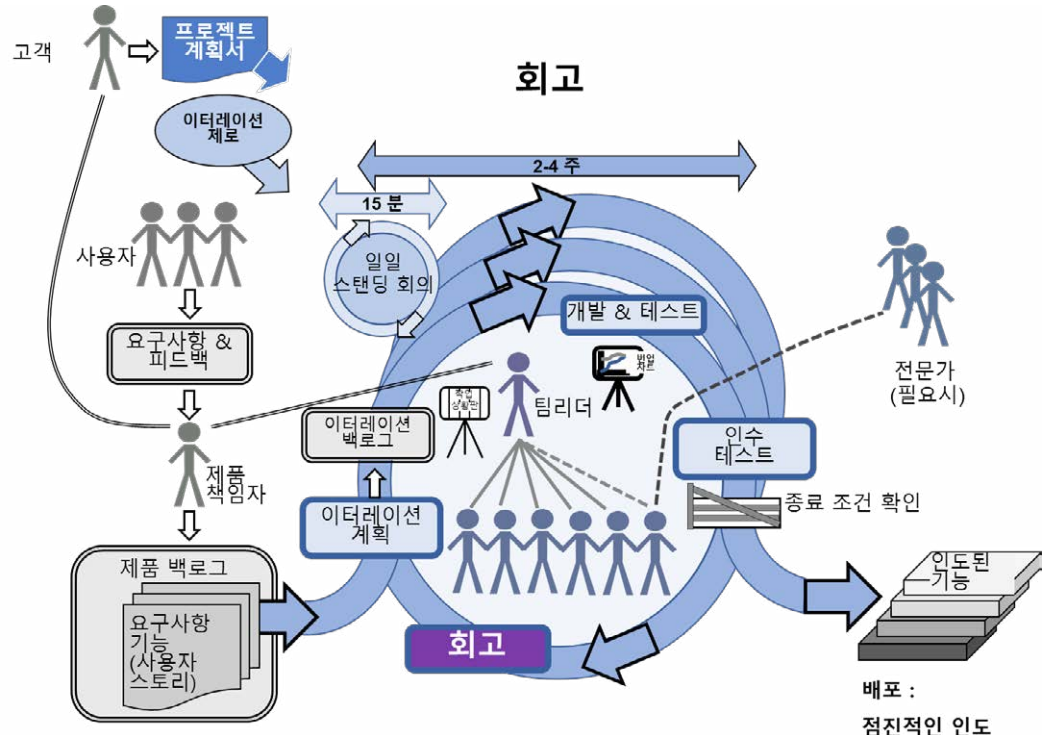
매 이터레이션이 종료될 때 가지는 미팅으로서 개발 프로세스(및 기타 지원 프로세스)에 대한 잠재적 개선 사항을 식별하고 합의한다.

■ 작업 산출물

- 없음

■ 역할

- 팀 (팀 리더 및 팀원)–3.1
- 제품 책임자–3.2



〈 그림 12. 구조도에서의 회고 미팅 〉

4.11.1 회고 시기

회고는 이터레이션의 종료 시에 수행되는 제한된 시간의 활동이다.

릴리즈 후에 회고를 수행해 릴리즈 계획의 성공 여부를 살펴보면서 릴리즈 내의 모든 이터레이션에 대해 돌아보는 것도 가능하다.

4.11.2 회고가 무엇인가?

회고는 팀과 제품 책임자가 완료된 이터레이션을 돌아보면서 프랙티스에 대한 개선 사항을 식별할 수 있는 기회이다.

4.11.3 회고 역할

회고의 참여자는 다음과 같다.

- **팀원, 전문가, 팀 리더 또는 (사용자를 대표하는) 제품 책임자.**
  - 잘된 것과 보다 잘될 수 있는 것에 대한 의견 제시
  - 작업 프랙티스의 개선을 위한 아이디어 제시
  - 다음 이터레이션에 이루어질 할 개선 사항에 대한 합의
- **팀 리더**
  - 미팅의 진행을 돕고 모든 사람에게 기회를 부여하고 사람들이 핵심에서 벗어나지 않게 하며 제한된 시간 내에 미팅 끝내기
  - 합의된 개선 사항이 구현되도록 하기

4.11.4 회고 작업 산출물

회고 미팅의 구체적인 작업 산출물은 없다. 개선을 위한 아이디어는 팀이 다음 이터레이션에서 구현하거나 제품 백로그의 항목이 되게 하기 위한 충분한 노력을 요할 수 있다. 일부 개선 사항(예: 외부로부터 지원되는 도구의 변경)은 팀 리더가 프로젝트 외부의 이해 관계자와 접촉할 것을 요구한다.

Tip 26. 회고에서 무엇을 기여할지 생각나지 않나요?

팀(과 제품 책임자)에 보다 도움이 될 개선 아이디어에 대해 생각해라.
- 가치 있는 소프트웨어의 인도
- 지속 가능한 작업
- 서로 및 사용자와의 원활한 의사 소통
- 기술적 우수성의 향상
- 보다 효과적일 수 있는 방안

예제 16. 회고 미팅

▶ 회고 미팅 수행하기

'마이트래블(MyTravel)' 웹사이트 프로젝트가 하나의 이터레이션을 끝냈고, 팀 리더인 '요한'의 주도 하에 회고 미팅이 진행되고 있다. 미팅의 다른 참석자들은 개발자인 '치원', 테스터인 '철순', UX 설계자인 '혜림'이다. 이들은 제품 책임자인 '숙자'를 초대했다.

먼저 이번 이터레이션의 잘된 점에 대해 논의한다. "'숙자'가 원하던 모든 사용자 스토리가 인도될 수 있었고 이번 인수 테스팅의 프로세스는 정말 원활하게 진행되었다." '철순'이 말한다. "그리고 사용자는 '혜림'의 인터페이스 설계에 만족했다! **가치 있는 소프트웨어를 인도**한 것이다."

"또한 이번 프로젝트와 다른 약속들 사이에서의 '숙자'의 시간적 갈등을 해결해 줄 수 있어서 정말 좋았다. **의사 소통이 개선**되었어요." '혜림'이 말한다.

그런 다음 이들은 잘되지 않은 점에 대해 이야기한다. '치원'이 한숨을 내쉬면서 말한다. "나는 모든 결함 보고에 다소 힘들었어요. '철순'은 훌륭한 테스터이고, 나는 당신이 하는 일에 정말 감사하고 있어요. 하지만 매일의 일과 끝에 보낸 모든 보고서를 처리하는 일이 때때로 힘들었어요. 스토리 테스팅이 시작된 후에는 매일 밤 추가 작업을 했어요." "오, 저런!" '철순'이 말한다. "나는 당신에게 힘든 일을 만들 생각은 없었어요!" '요한'이 말한다. " 그것을 개선할 수 있을까요? 그것이 '치원'에게 **지속 가능하지 않고 효과적이지도 않다는** 것을 알 수 있잖아요."

'철순'이 고개를 끄덕이면서 말한다. "오류 보고에 관해 말하자면, 나는 모든 것을 기록하고 하루에 한 번만 그 목록을 주는 것이 '치원'에게 도움이 된다고 생각해. 이 일을 보다 비공식화해야 할지 생각해. '요한', 당신이 시작할 때 비공식적 결함 관리를 제안했지만 내가 꺼렸었는데, 지금은 스토리 테스팅 동안 '철순'과 보다 긴밀하게 작업한다면 더 나을 것으로 생각되는군요." '요한'이 미소를 짓는다."여러분은 어떻게 생각하세요?"'치원'이 말한다. "'철순'이 오류를 발견하는 즉시 내게 준다면 나는 하고 있던 일을 계속 잊어 버리게 될 것이지만 오류에 대해 보다 빨리 들어서 어떤 것이 가장 중요한지 알고 싶어요. 그리고 매우 중요한 경우라면 늦게까지 남아서 수정해 최대한 빨리 테스트 받게 하고 싶어요." '혜림'이 제안한다. "'철순'이 오류들을 발견해 나에게 넘겨 주면, UX에 그것들이 얼마나 중요한지 살펴본 후에, 우선 순위대로 '치원'에게 넘겨주면 어떨까요?" '치원'이 큰 소리로 말한다. "아니면, 오류를 발견했을 때 기록하는 대신 작업상황판에 올려 놓으세요. 그러면 내가 현황판에서 그것들을 확인해 준비되는 대로 처리할 수 있을 겁니다." '철순'이 동의한다. "내가 붉은 색 포스트잇을 사용하면 명확히 볼 수 있을 겁니다." '치원'이 대답한다. "**보다 효과적**이겠군요! 하지만 **지속 가능**할까요?"

"'철순', 세션에 기초한 접근법을 사용하면 된다." '요한'이 제안한다. "2시간 동안 테스트한 후 '혜림', '치원'과 20분 동안 만나서 테스트 결과에 대해 논의하고 수정돼야 할 점을 합의하면 된다." '철순'이 고개를 끄덕이며 말한다. "네, 오늘의 마지막 만남에서 내일 수정돼야 할 것이 정해진다면 '치원'이 늦게까지 머무를 필요가 없네요. 그렇게 하면 **지속 가능**할 겁니다. 나로서도 해당 이터레이션에 해결할 계획이 없는 오류만 기록함으로써 **보다 효과적이고 지속 가능**하게 작업할 수 있어요."

'치원'이 말한다. "나는 결함들 중 많은 부분이 회귀 문제라고 생각한다. 회귀에 있는 스토리 테스트들의 일부를 자동화해 단위 테스트 동안 수행한다면, 문제를 보다 빨리 발견해 **기술적 우수성을 향상**시키게 될 겁니다." '요한'과 '철순'이 동의한다. "좋아요. 이 모든 것을 다음 이터레이션에 시도해 봅시다." 참석자 모두가 동의한다.

4.12 운영 및 유지보수

운영 및 유지보수

■ 목적

소프트웨어가 인도된 후 사용자가 자신의 작업을 수행할 수 있도록 지원하기

■ 설명

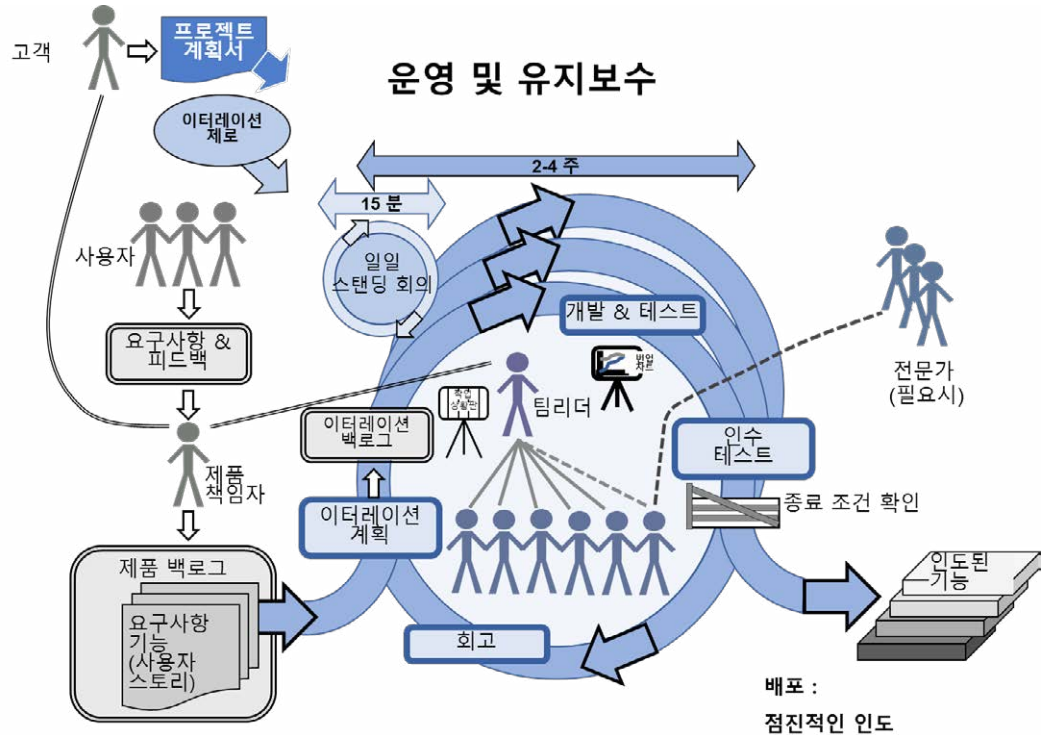
일단 소프트웨어가 사용자에 의해 사용되도록 인도되면 그 시스템 사용의 지원에 대한, 그리고 시스템 변경에 대한 요구 사항이 있다. 예를 들면, 결함(defects)를 해결하거나 향상에 대한 요구 같은 것이다.

■ 작업 산출물

- 동작하는 소프트웨어

■ 역할

- 사용자–3.6
- 제품 책임자–3.2
- 팀(팀 리더 포함)–3.1



〈 그림 13. 구조도에서의 운영 및 유지보수 〉

4.12.1 영 및 유지보수 시기

운영 및 유지보수는 첫 소프트웨어가 사용자에게 릴리즈될 때 시작되고 해당 시스템이 더 이상 사용되지 않을 때 종료된다.

4.12.2 운영 및 유지보수가 무엇인가?

운영 및 유지보수의 목적은 사용자를 지원하고 시스템을 유지보수(예: 결함 해결, 시스템 향상)하는 것이다. 프로젝트의 초기 개발 단계가 완료되면 해당 소프트웨어가 배포돼 사용된다. 하지만 사용자는 여전히 소프트웨어에 대한 변경을 원할 것이다. 이러한 유지보수 변경 사항들은 다음과 같다.

- 새로운 기능의 추가
- 소프트웨어의 비기능적인 속성 개선
- 사용중인 플랫폼의 변경
- 데이터 이동
- 결함 해결

팀은 프로젝트 전반에 걸쳐 사용해 온 방식대로 방법론을 사용하면서 사용자에게 계속 서비스할 수 있다. 운영 및 유지보수에는 초기 개발 단계에서와 동일한 접근법이 사용된다.

운영 및 유지보수는 이미 존재하는 프로젝트 계획서의 맥락에서 수행될 수 있다. 이 경우 아무 것도 변경되지 않으며 팀은 전과 같이 제품 책임자와 협력해 이터레이션을 수행한다.

때로는 프로젝트에 대한 조직의 목표가 변경돼 프로젝트 계획서를 재정의해야 할 수도 있다.

4.12.3 운영 및 유지보수 역할

운영 및 유지보수 역할은 프로젝트의 개발 및 테스팅 부분의 역할과 동일하다.
프로젝트에 대한 '전제 조건'에서 보았듯이 프로젝트는 여러 관련 프로그램의 일부로서 매우 자주 수행된다. 팀이 처음 계획된 소프트웨어의 릴리즈를 인도한 후에 고객은 팀이 새 프로젝트에 대해 작업해 주기를 원할 수 있다. 이것은 팀이 이전 프로젝트를 기반으로 한 집단의 사용자를 위해 운영 및 유지보수를 수행하면서, 동시에 다른 제품 책임자 및 사용자와 함께 새 개발 프로젝트를 수행하는 것을 의미한다. 이 경우 팀 리더는 두 제품 책임자의 요구들 사이에서 팀의 시간이 합리적인 방법으로 분할되도록 하기 위해 두 제품 책임자와 협의해야 한다.

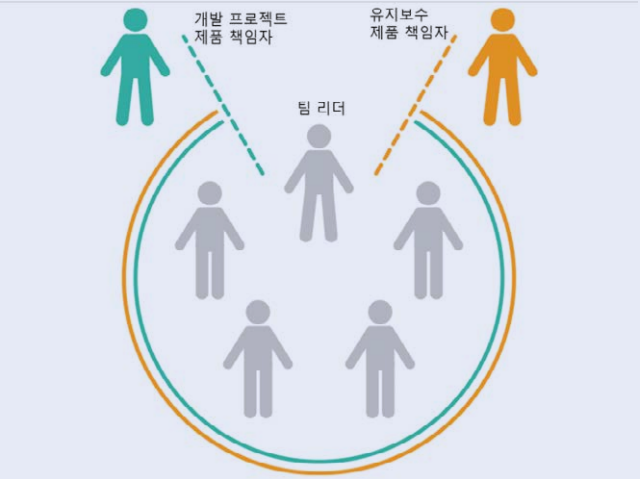
Tip 27. 개발 및 유지보수 작업?

이상적으로는 팀(팀 리더 및 팀원)이 한 번에 하나의 개발 프로젝트에 대해서만 작업하면서 배포된 시스템에 대한 유지보수를 수행하도록 비정기적으로 요구 받아야 한다. 그러나 때로는 하나의 프로젝트에 대한 운영 및 유지보수가 다른 프로젝트의 개발과 병행해 수행된다. 팀 리더는 팀의 시간이 담당 작업들 사이에 잘 분할되도록 하기 위해 두 제품 책임자와 협의해야 한다.

예제 17. 개발 및 유지보수 작업

▶ 팀 리더가 하나의 팀으로 두 명의 제품 책임자와 함께 두 개의 프로젝트를 수행하고 있다.

'나리'는 하나의 팀을 이끌면서 두 개의 프로젝트에 대해 작업하고 있다. 즉, 한 명의 제품 책임자를 위한 유지보수 프로젝트에서 계속 기존 시스템을 업데이트하면서, 다른 한 명의 제품 책임자를 위한 새 응용프로그램을 개발하고 있다. 팀 전체가 두 프로젝트에 대한 작업을 수행하고 있으며 '나리'는 시간을 공유하는 방법을 도출하기 위해 두 제품 책임자와 협의한다.



4.12.4 운영 및 유지보수 작업 산출물

운영 및 유지 보수 작업 산출물은 프로젝트의 개발 및 테스팅 부분과 동일하다. 자세한 내용은 7. 운영 및 유지보수 수행 활동을 참조하라.

CHAPTER 05

프로젝트 수행 상세 방법

CHAPTER 05 프로젝트 수행 상세 방법

5.1 이터레이션 제로

5.1.1 이터레이션 제로 스타트업 미팅

최초의 프로젝트인 경우 방법론 코치가 있으면 좋다. 필요한 경우 제품 책임자, 팀 리더 및 팀을 위한 이터레이션 제로에 대한 교육 시간이 필요할 수 있다.

5.1.2 팀원 모집

필요한 모든 팀 기술을 결정하고 모집에 합의한다. 이것은 결정된 팀 규모(팀 리더를 포함해 최소 3명, 최대 10명)와 다양한 기술 또는 전문적 자원의 필요 여부에 기반한다. 팀에서의 역할들을 확인해 명확한 팀 구조를 만든다. 그런 다음 팀원을 내부에서, 외부에서 혹은 둘 다로부터 모집할 지를 결정한다.

5.1.3 자원

팀 규모와 전문가의 참여에 따라 다음을 수행한다.

- 사무실 공간, 데스크, 테이블 및 장비를 구성해 팀이 공동 작업할 수 있게 한다.
- 필요한 IT 인프라, 도구, 환경을 준비한다.
- 보안(IT, 장비, 정보 액세스)을 준비한다.
- 관련이 있는 경우 특수 테스트 장비, 관련 표준, 특수 도구, 접근성 요구 사항 같은 필요한 자원을 식별한다.

팀 공간은 공유되는 자원이 있는 하나의 방 또는 큰 방의 일부일 수 있다. 다른 팀의 방해와 소음으로부터 팀을 보호할 수 있는 공간을 설계하라. 많은 토론을 위해 필요하다. 팀 리더는 별도의 사무실이 없다. 팀 리더는 팀원과 함께 작업한다. 팀 전체를 위한 충분한 작업 공간이 있는지 확인. 화이트보드 및 작업상황판과 릴리즈 번-업 차트를 위한 게시판이 있는지 확인. 명확히 구분된 토론을 위한 영역과 조용한 작업을 위한 영역이 있는지 확인.

Tip 28. 자원 계획 (Planning resources)

최소한의 도구 및 기타 자원으로 시작하고 다른 것이 필요한 경우를 위한 계획을 세워라. 예를 들면

- 매 이터레이션마다 인수 테스팅이 수행될 예정인 경우 이터레이션1부터 인수 테스트 환경이 필요하다.
- 인수 테스팅이 덜 빈번하게 수행될 예정인 경우 인수 테스트가 수행되는 이터레이션까지는 인수 테스트 환경이 필요하지 않다.

5.1.4 아키텍처 및 기술적 결정

아키텍처와 설계에 사용되는 방법은 조직이 선호하는 소프트웨어 엔지니어링 방법과 직면한 문제에 따라 다르다. 그 예는 다음과 같다.

- UML
- 사용자 중심 설계
- 기능 주도 개발방법론(Feature-Driven Development)

5.1.5 초기의 제품 백로그 관리 및 릴리즈 계획

제품 책임자는 다음을 수행한다.

- 고객의 목표와 사용자의 필요를 충족시키는 사용자 요구 사항 목록을 작성한다.
- 사용자 및 고객에 대한 다른 요구 사항들을 토론에 참고한다.
- 소프트웨어의 배포 빈도에 관해 사용자 및 고객과 합의한다.
- 인수 테스팅의 빈도에 관해 사용자와 합의한다.
- 배포의 계획된 소프트웨어와 빈도를 보여주는 릴리즈 계획서를 작성하고 인수 테스트가 수행될 때 이것을 팀 리더와 함께 검토한다.
- 우선 순위가 가장 높은 요구 사항에 대해 사용자 스토리 카드를 작성한다.

Tip 29. 첫 번째 제품 백로그

첫 프로젝트인 경우 다음이 권장된다.

- 현재의 릴리즈 패턴(존재할 경우)으로부터 개선할 수 있는 릴리즈 계획을 세운다.
  과거에 소프트웨어가 매년 릴리즈되었다면 첫 개선은 분기별로 4회 릴리즈를 수행하는 계획을 세운다.
  새 프로젝트에 대해 사용자 및 제품 책임자는 매달 인수 테스팅 및 배포를 수행한다.
- 제품 백로그 및 릴리즈 계획에 추가하는 작업에 대해서는 사용자와 공동 작업할 수 있도록 합의하라.

5.1.6 완료 정의

"완료(Done)"라는 용어는 이터레이션 동안 완료된 작업의 일부, 배포 준비가 된 사용자 스토리의 완성, 또는 배포될 인도 가능한 전체 소프트웨어 세트를 지칭하는 말이다. 고객, 제품 책임자, 팀 리더, 팀 및 사용자 모두가 하나의 공유된 완료 정의를 가지는 것이 중요하다.

첫 프로젝트인 경우 다음이 권장된다.

- 각 활동에 대해 "미니 완료"를 정의하고 이 활동을 작업상황판의 열과 일치시키라. 또한, 사용자 스토리를 한 열에서 다음 열로 이동하는 일은 '미니 완료'의 정의를 충족시키는 지에 따른다.
- 배포가 완료되었음을 의미하는 초기 완료 정의를 하라.

예제 18. 완료 정의 (Definition of Done)

▶ 제품 전시회 준비하기 – "완료"를 위해 필요한 것은 무엇일까?

'릴리(Lily)' 프로젝트는 7월 제품 전시회에서 릴리즈할 게임을 제작 중이다. 고객은 영업 및 마케팅 책임자인 '고은'이다. 제품 책임자는 '보람'이다. 이것은 소매업체에 공급돼 일반인에게 판매될 상용 소프트웨어이다. '보람'과 '고은'은 이터레이션 제로 동안 릴리즈 계획에 합의했다.
'보람'은 7월 릴리즈 전에 여덟 번의 4주 이터레이션을 갖는 릴리즈 계획에 동의했다. 세 번의 중간 인수 테스트는 매 두 번째 이터레이션마다 수행될 것이며 최종 전체 인수 테스트는 여덟 번째 이터레이션이 끝날 때 수행될 것이다.
'보람'과 팀 리더는 7월 릴리즈 이전에 각 이터레이션 내 활동, 이터레이션, 중간 인수 테스트 및 전체 인수 테스트를 위한 완료 정의에 합의한다.

이터레이션 내 활동:

– 설계 완료 – 검토되고 합의되었음을 의미한다.
– 개발 완료 – 코드가 작성돼 정적 분석을 통과하고 100% 문장 커버리지(statement coverage)으로 단위 테스트도 통과했으며 자동화된 단위 회귀 테스트도 통과했음을 의미한다.
– 스토리 테스트 완료 – 탐색적 테스팅이 완료되고 스토리 회귀 테스팅도 통과했으며 비 기능 테스트도 통과했음을 의미한다.

매 두 번째 이터레이션:

– 중간 인수 완료 – 판매 및 마케팅 담당자가 게임을 실행하였고 접근성 및 사용성 테스트를 통과했음을 의미한다.

최종 인수 테스트:

– 고객 대표가 인수 테스트를 수행한다.
– 모든 인수 테스트를 통과한다.
– 모든 회귀 테스트를 통과한다.
– 모든 참여자가 만족도 설문지를 작성하고 그 결과가 인수 수준을 통과한다.

완료된 활동	내용	담당자
설계 완료	설계 검토 및 합의	팀 수석 아키텍트
개발 완료	작성된 코드 정적 분석 통과 100% 문장 커버리지(statement coverage)로 단위 테스트 통과 자동화된 단위 테스트 단위 회귀 테스트 수행 및 통과 자동화된 회귀 스토리 테스트 수행 및 통과	개발자
스토리 테스트 완료	탐색적 테스팅 완료 수동 회귀 테스트 수행 및 통과 회귀 테스트 자동화 완료 비기능 테스트 수행 및 통과 결함 해결 (제품 백로그에 대한 수정/재테스트)	테스터
인수 테스트 완료	인수 테스트 세트 수행 및 통과 인수 회귀 테스트 수행 및 통과 인수 기준 충족 결함 해결 (제품 백로그에 대한 수정/재테스트)	제품 책임자
배포 완료	데이터 이동 소프트웨어 실행 소프트웨어 사용	팀 리더
완료	설계 완료 개발 완료 스토리 테스트 완료 인수 테스트 완료 배포 완료	제품 책임자 / 팀 리더

[산출물 1. 완료 정의]

Tip 30. 완료 정의 변경

프로젝트를 수행한 후 개발 및 테스트 프로세스에 대한 경험 및 개선 사항을 기반으로 향후 프로젝트에 대한 완료 정의를 변경하고 싶을 수 있다. 예를 들어 위의 표에서 '개발 완료'에 다음을 추가할 수 있다.

- 사용자 스토리에 대한 단위 테스트 성공적으로 통과 (모두 통과)
- 필요한 경우 코드/설계 리팩터링
- 동료 검토 통과
- 새 코드가 빌드에 통합됨
- 작업상황판을 업데이트하고 모든 팀원에게 전달

완료 정의가 회귀 테스팅의 완료를 포함하고 있음을 항상 확인

5.1.7 초기 프로세스 정의

첫 프로젝트인 경우 이 가이드의 원리 및 권장 사항을 조직에 맞게 잘 적용하라.

프로젝트를 이미 수행했다면 개선 사항을 터득해 적용했을 것이고 상황에 맞는 가이드도 작성했을 것이다. 이터레이션 제로에, 팀원, 팀 리더, 제품 책임자 및 전문가는 이전 프로젝트에서 유용한 것으로 판단된 개선 사항을 논의해 해당 프로젝트에 적용할 기회를 가질 수 있다.

이러한 논의에서 현재의 프로젝트를 위한 초기 프로세스 정의에 대해 합의하라.

5.1.8 추정 준비

• 이터레이션 계획 준비 •

각 이터레이션에 팀이 관리할 수 있는 것보다 더 많은 작업을 무리하게 해서는 안 된다. 작업량이 매 이터레이션마다 변동돼 지속 불가능해져서 팀이 스트레스와 질병으로 실패하게 될 것이다. 이터레이션 계획 동안 추정해 지속 가능한 페이스를 유지하려면 다음을 파악해야 한다.

- 해당 프로젝트를 위한 이터레이션 기간
- 해당 이터레이션에 개발 작업(설계, 코딩 및 테스팅)에 사용 가능한 날의 수와 인수 테스팅에 사용 가능한 날의 수
- 해당 이터레이션에 계획 및 회고에 소요되는 시간
- 스토리 포인트가 의미하는 것
- 팀의 "속도" – 해당 이터레이션에 완료할 수 있는 작업량

이터레이션 제로 동안 이러한 모든 요소들의 출발점을 평가하고 합의해 이터레이션 계획을 준비한다.

• 이터레이션 기간 결정 •

하나의 이터레이션은 최대 4주이다. 이터레이션 기간이 프로젝트 동안 변경돼서는 안 된다.
제품 책임자, 사용자 및 고객과 인수 테스트 및 배포 간격에 대해 합의했거나 릴리즈 계획이 있는 경우 이터레이션 기간을 그것에 맞추는 것이 좋다. 마음 편히 인도할 시간을 확보할 수 있도록 이터레이션 기간을 정하라.

예제 19. 이터레이션 기간의 결정

▶ 사용자 및 팀에 맞추어 이터레이션 기간 정하기

'마이트래블(MyTravel)'의 사용자인 '여진'은 너무 바빠서 매 이터레이션마다 인수 테스트를 할 수 없다. 그녀는 매 3개월마다 인수 테스트와 배포가 수행되기를 원한다. 매 3번째 이터레이션에 전체 비기능 테스트가 포함된 인수 테스트 및 배포를 수행하는 것과 함께, 4주 단위의 이터레이션을 갖는 것이 합리적이다. '요한'과 '숙자'는 4주 단위 이터레이션 패턴을 위한 '마이트래블(MyTravel)' 프로젝트 릴리즈 계획을 작성한다.

Tip 31. 이터레이션 기간 결정하기

사용자가 매 이터레이션마다 인수 테스트와 배포를 원하는 경우, 이터레이션의 마지막 주에 인수 테스팅, 필요한 수정 및 배포를 수행하는 4주 단위 이터레이션으로 시작하는 것이 합리적일 수 있다.

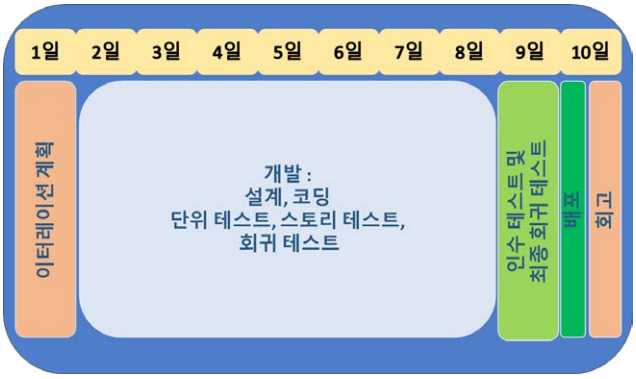
프로젝트가 작고 리스크가 낮은 경우, 사용자와 팀 사이의 긴밀한 작업 협력 및 의사 소통을 통해 이터레이션의 마지막 날 또는 마지막 2일 동안 인수 테스트와 배포를 수행하는 1~2주 단위의 이터레이션으로 시작하는 것이 현명할 수 있다.

• 사용 가능한 이터레이션 일수 – 이터레이션 제로 동안 평가 •

이터레이션 제로 동안, 한 이터레이션의 기간, 이터레이션 시작 시에 이터레이션 계획을 위해 부여되는 시간, 이터레이션 종료 시에 인수 테스팅, 배포 및 회고 미팅을 위해 부여되는 시간에 합의하라.
한 가지 예가 아래에 제시돼 있다.

• 개발 작업을 위해 몇 일이 이용 가능한가요? •

'PIP' 팀은 2주 이터레이션 동안, 이터레이션 계획에 1일, 회고 미팅에 0.5일, 인수 테스팅, 최종 회귀 테스팅 및 배포에 1.5일을 할당 받았다. 이것은 설계, 코딩, 단위 테스팅, 스토리 테스팅 및 회귀 테스팅을 포함한 개발 작업을 위해 7일이 할당됨을 의미한다.



[도표 1. 이터레이션 10일 계획]

• 스토리 포인트에 대해 합의하기 – 이터레이션 제로 동안 평가 •

스토리 포인트를 사용해 다음을 설명한다.

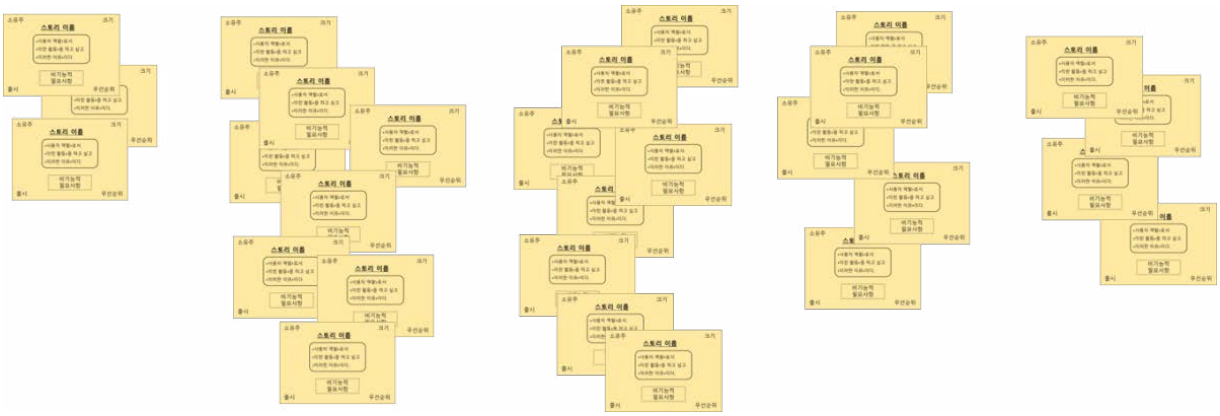
- 스토리의 크기
- 스토리 완료를 위한 작업량
- 이터레이션 완료를 위한 작업량
- 릴리즈 완료를 위한 작업량
- 팀의 속도 (팀이 하나의 이터레이션에 인도할 수 있는 스토리 포인트 수)

스토리 포인트에 대해 기억해야 할 정말 중요한 점은 이것이 상대적인 척도라는 것이다. 즉, 어떤 것을 완료하는 데 몇 시간 또는 몇 일이 걸릴지를 의미하지 않는다. 스토리 포인트는 스토리 크기에 대해 표현할 수 있는 하나의 방법이다. 이터레이션이 지나면서 스토리의 크기를 보다 정확하게 추정할 수 있다. 스토리 포인트는 상대적인 척도를 제공하기 때문에 다른 팀은 다른 크기를 생각할 수 있다. (한 팀은 3주 이터레이션 동안 300 스토리 포인트의 스토리를 인도할 수 있고, 다른 팀은 같은 기간 동안 60 스토리 포인트의 스토리를 인도할 수 있다. 아래에서 알 수 있듯이 점수를 할당하는 방법에 따라 다르다.)
그러면 새 팀은 이 척도에 대해 어떻게 합의할까? 팀은 프로젝트를 위한 스토리 포인트 하나의 크기에 대해 합의해야 한다.
이터레이션 제로 동안 이 작업이 수행될 것이므로 이터레이션1에 첫 이터레이션 계획 미팅을 시작할 때 추정을 시작할 준비가 돼 있다.
이를 위해 초기 제품 백로그가 필요하다.

Tip 32. 아직 전념하지 않고 있음: 스토리 포인트 평가만 하기

위에서 이터레이션 제로 동안 사용자 스토리 카드들에 최우선 순위를 정하는 것을 포함한 초기 제품 백로그 관리 활동을 제품 책임자가 수행할 것임을 보았다. 제품 책임자는 사용자가 향후 3 개월 동안 필요하다고 말한 것에 대해 생각할 것이다. 팀은 아직 그 작업에 전념하고 있지 않으며 가능한지의 여부조차 모른다. 이 시점에 스토리 포인트가 의미하는 것을 시도하고 합의하기 위해 카드 세트를 사용할 것이다.

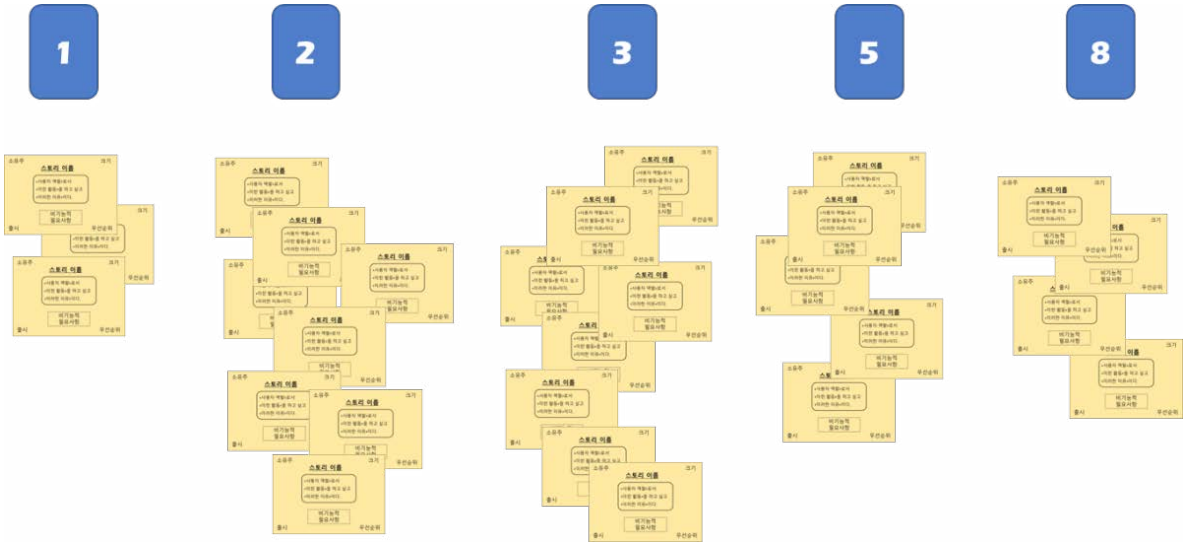
팀은 각각의 스토리를 차례로 가져와서 그것의 설계, 코딩, 스토리 테스팅 및 회귀 테스팅에 소요될 시간을 대략적으로 추정하고 합의를 도출한다. 스토리들은 5 그룹으로 분류되고 스토리들의 상대적 크기가 가장 작은 것에서 가장 큰 것까지 표시된다. 잠시 토론을 거친 후 카드들이 다음처럼 보이도록 한다.



[도표 2. 스토리 포인트 평가하기 – 초기의 카드 분류]

그 그룹에 새로운 스토리를 추가하는 동안 어떤 스토리가 잘못된 그룹에 있다고 판단될 경우 그 스토리를 올바르다고 합의된 그룹으로 옮긴다. 가장 큰 스토리도 구현하는 데 5일 이상 걸리지 않아야 한다. 너무 커서 5일 이상 걸릴 경우 보다 작은 스토리들로 나누어야 한다.

스토리 포인트는 피보나치 수열(Fibonacci sequence)의 처음 5개 숫자, 즉, 1, 2, 3, 5, 8에 기반한다. 다음으로 5개의 그룹에 라벨을 추가한다. (가장 작은 스토리를 위한 1부터 가장 큰 스토리를 위한 8까지) – 이 숫자들은 스토리의 상대적 크기를 나타낸다.



[도표 3. 스토리 포인트 평가하기 – 스토리 포인트에 라벨 추가하기]

이제 라벨에 있는 상대적 크기를 인식하면서 카드 그룹을 다시 검토해 일부 카드를 다른 카드 그룹으로 옮길 지 결정할 수 있다. 작업을 마무리하기 전에 예를 들어, '2'로 표시된 그룹의 스토리는 '1'로 표시된 그룹의 스토리를 구현하는 시간의 약 두 배 정도 걸려야 하며, '8'로 표시된 그룹에 있는 스토리들보다는 훨씬 적은 시간이 걸려야 한다는 것에 합의해야 한다. 하지만 팀원은 정확히 추정할 수 없을 수도 있다.

Tip 33. 스토리 포인트 크기에 대해 합의하기

예를 들어 '6'이 돼야 할 스토리가 있는데 '6'이라고 표시된 그룹은 없다고 염려하지 마라. 피보나치 수열을 사용하는 이유는 5 크기의 일 다음에는 6보다는 8 정도 크기의 일이 될 가능성이 높기 때문이다. 팀원은 1에서 8까지의 전체 범위가 필요한 것을 추정하는 데 그리 능숙하지 않으며 정확하기 위해 너무 오래 시간을 들이지 않아야 함을 잊지 마라.

서로 다른 팀들은 서로 상당히 다르게 점수를 매길 것이라고 위에서 말한 것을 기억하라. 왜냐하면, 어떤 팀은 약간 다른 피보나치 숫자(예: 2, 3, 5, 8, 13)를 사용하기로 결정할 수 있고, 어떤 팀은 1, 2, 4, 8, 16, 32(5개 그룹에만 국한되지 않음에 주목하라)의 베이스-2 수열을 사용할 수 있기 때문이다. 이 점수들은 상대적일 뿐이며 시간이나 일수를 나타내지 않는다는 것을 다시 한 번 기억하라.

연습을 마쳤으므로 이제 스토리의 크기를 정하는 방식에 대한 기준을 갖게 되었다.
물론 이렇게 추정하는 것이 서투르고 스토리의 상대적 크기에 대한 '추측'이 다소 잘못돼 있을 수 있지만 경험이 쌓이면 향상될 것이다.

• 가능한 속도(velocity)에 대해 합의하기 – 이터레이션 제로 동안 평가 •

매 이터레이션에 인도할 수 있는 소프트웨어를 추정하려면 팀의 속도를 알아야 한다. 속도는 팀이 하나의 이터레이션에 인도할 수 있는 작업량이다. 프로젝트가 진행되면 이전 3회의 이터레이션 동안 인도한 평균 양을 검토해 속도를 평가한다. 하지만 프로젝트가 시작될 때는 그러한 정보를 가지고 있지 않다. 이터레이션 제로 동안 하나의 출발점으로서 추정되는 속도를 설정해야 한다.
이 시점에서는 스토리 포인트를 사용해 결정하는 대신 인도할 수 있다고 생각되는 것에 대한 최상의 추측만 사용하라. 과대 평가해서는 안 된다. 이전의 경험에 기초해 토론해서 가능한 속도에 대한 합의에 이를 것이다. 이렇게 출발점을 제공한다.

5.1.9 팀 작업실 설치

좋은 팀 작업실에는 다음이 포함된다.

- 편안한 토론 및 미팅을 위한 장소
- 일반적인 작업을 위한 장소
- 개인적 미팅이나 집중적인 작업을 위한 조용한 장소
- 작업상황판, 화이트보드, 프로젝트 계획서, 번-업 차트 같은 게시물이 있음
- 보관 장소
- 팀이 필요로 하는 장비가 구비됨

5.1.10 프로젝트 시작 미팅

프로젝트 시작 미팅에는 다음 두 부분이 포함된다.

우선 모든 사람(고객, 사용자, 제품 책임자, 팀 리더, 전문가 및 팀원)이 참석하며 제품 책임자가 회의를 주제한다.

- 릴리즈 계획과 이터레이션 기간을 소개하면서 모든 사람들이 파악하고 있는지 확인한다.
- 소개 – 모두가 서로를 알고 있는지 확인한다.
- 모든 사람을 환영하며 감사를 표한다.
- 모든 사람이 프로젝트 계획서를 파악하고 있는지 확인한다. 즉 프로젝트의 목표(목표, 제약 조건, 대상, 성공 지표)가 명확해야 한다.
- 역할과 책임 – 모든 사람이 서로가 수행하는 작업, 보유한 기술, 담당할 부분을 알고 있는지 확인한다.
- 모든 사람이 정보가 있는 곳과 합의된 의사 소통 수단을 알고 있는지 확인한다.
- 고객 및 사용자에게 감사하고 미팅의 첫 번째 부분을 마칩니다.

그 후 팀과 제품 책임자가 미팅을 계속 진행한다. 고객 및 사용자는 반드시 필요하지는 않다.

- 상세한 작업 방법에 대해 합의한다.
- 이터레이션1의 준비 여부를 확인한다.
- 프로젝트에 전념할 것을 약속한다.

5.2 제품 백로그 도출 및 관리

5.2.1 제품 책임자 책임

제품 책임자는 이터레이션 계획이 작성될 때 가장 중요한 요구 사항이 제품 백로그에 사용자 스토리로 표현되도록 할 책임이 있다. 다음 3회의 이터레이션에 작업하기에 충분할 만큼의 사용자 스토리들만 있으면 된다. 더 이상을 준비하는 것은 우선 순위와 요구 사항의 변경으로 인해 결코 구현되지 않을 사용자 스토리를 작성하는 데 노력을 낭비하는 일일 뿐이다.

다음 이터레이션 계획 미팅을 위해 우선 순위가 가장 높은 스토리들과 다음에 구현되기 원하거나 구현될 필요가 있는 것에 대해 알고 있을 것이다. 스토리에는 일반적으로 다음 네 가지의 주요 유형이 있음을 기억하라. 사용자 요구 사항을 표현하는 스토리, 다른 스토리들이 의존하기 때문에 구현돼야 하는 스토리, 보다 효율적인 작성을 위해 팀이 기술적 작업을 수행해야 할 스토리 및 해결돼야 할 결함(defects)를 표현하는 스토리.

스토리들의 우선 순위를 정하는 동안 제품 백로그에 있는 스토리들 중 일부가 시간이 지남에 따라 대체되었거나 이제는 진부하거나 단순히 잘못돼 있음을 발견하게 될 것이다. 이들을 제거해 향후에 백로그 관리가 수월해지도록 하라.

5.2.2 사용자 요구 사항 식별하기

제품 책임자는 시스템에 대한 새로운 요구 사항이나 변경된 요구 사항이 있는지 정기적으로 사용자에게 확인해야 한다.

조직에 따라 다음을 수행한다.

- 완료된 사용자 스토리를 보내도록 사용자에게 요구한다.
- 개별적으로 또는 필요에 따라 사용자와 대화할 수 있는 기회를 마련하거나 사용자가 (E-메일 또는 다른 형식으로) 요구 사항을 보내도록 한다.
- 사용자와의 미팅을 가지면서 원하는 것을 질문한다.

5.2.3 사용자 스토리 우선 순위 정하기

사용자가 요구 사항의 긴급함이나 해결해야 할 결함의 심각성을 알릴 수 있는 방법에 대해 사용자와 합의하는 것이 중요하다. 예를 들면

예제 20. 요구 사항에 대한 긴급함의 정도 정하기

▶ 요구 사항이나 결함의 긴급함을 문서화한 예

! = 응급 상황!!!
A = 다음 이터레이션이 끝나기 전에 **반드시** 완료돼야 한다.
B = 다음 이터레이션이 끝나기 전에 완료되기 **바랍니다.**
C = 다음 3회의 이터레이션 동안 완료해 주세요.
D = 나중에 수행해도 된다.

5.2.4 팀 요구 사항

팀으로부터의 기술적 스토리 요구 사항은 두 가지 방식으로 발생할 가능성이 크다. 이터레이션 계획 중에 사용자 스토리들 중 하나에 추가적인 기술 작업이 필요하다는 것을 팀이 인식하거나 회고 동안에 도구 세트를 개선할 필요가 있음을 인식한다. 이러한 요구 사항을 평가할 때 팀의 보고를 신중하게 경청하고 팀 리더와 협력해 대안이 있는지 결정하며 단기 및 장기의 사용자에게 무엇이 최상의 가치를 제공할 것인지에 초점을 맞춘다.

5.2.5 제품 백로그 미팅

제품 백로그 미팅은 다음 이터레이션 계획 미팅을 준비하기 위해 이터레이션별로 열리거나 보다 비정기적으로 수행될 수 있다. 이상적으로는 사용자와 팀 모두를 불러서 제품 백로그에 있는 내용을 파악하도록 하고 새 요구 사항을 추가하고, 더 이상 필요하지 않은 요구 사항을 제거하고, 다음 몇 번의 이터레이션을 위한 프로젝트 방향을 파악하는 것이다.

5.2.6 사용자 스토리 작성하기

사용자 스토리를 직접 작성하거나 이해 관계자(사용자 및 팀원)가 스토리를 작성할 수 있다. 사용자 스토리가 항상 세부 사항을 바로 알 수 있는 스토리 카드에 기록돼 있는지 확인.

Tip 34. 정기적인 백로그 미팅

사용자와의 정기적인 백로그 미팅은 분기별로 열릴 수 있다. 각 사용자의 요구 사항들이나 우선 순위들이 상충하는 경우 도움이 될 수 있다. 제품 책임자는 사용자가 다음 분기에 달성하기 원하는 것을 사전에 논의한다. 제품 책임자는 기술적 요구 사항이 있는 경우 팀과도 대화한다. 회의 시작 시에 제품 책임자는 “이번 분기에 수행”, “12개월 동안 수행”, “나중에 수행” 및 “보류”의 라벨이 붙은 각 테이블 위에 사용자 스토리 카드 그룹을 펼쳐 놓는다. 사용자 및 팀원은 제품 책임자와 협력해 다음 분기를 위해 계획된 이터레이션에 “이번 분기에 수행” 카드 그룹을 수행하는 것이 가능할지 확인한다. 이 미팅은 릴리즈 계획과 연결될 수도 있다. 또한 팀이 프로젝트를 수행하는 방식에 있어서 사용자가 알고 싶어하는 개선 사항에 대해 논의할 기회이기도 한다.

예제 21. 정기적인 백로그 관리

▶ '릴리(Lily)' 프로젝트를 위한 정기 백로그 미팅이 제품 전시회까지 이어진다.

'릴리(Lily)' 프로젝트는 7월 제품 전시회에서 릴리즈될 게임을 제작 중이다. 고객은 영업 및 마케팅 책임자인 '고은'이다. 제품 책임자는 '보람'이다.

이것은 소매업체에 공급돼 일반인에게 판매될 상용 소프트웨어이다. '보람'과 '고은'은 이터레이션 제로 동안 릴리즈 계획에 합의했다. 백로그 계획 및 인수 테스팅을 위해 영업, 마케팅 및 지원 팀의 사람들이 "사용자"를 대표할 것이다. 왜냐하면 이들은 가장 가까이에서 사용자를 파악하고 있기 때문이다.

지금은 11월이며 '보람'은 7월 릴리즈할 방법을 계획하고 있다. 8회의 4주 단위 이터레이션이 수행될 것이며 론칭을 준비하기 위해 회사 전체 및 주요 고객사들이 참가하는 대규모 인수 테스트가 최종 이터레이션에 수행될 것이다.

'보람'은, 판매, 마케팅 및 지원 팀의 대표자 1명 또는 2명과 함께 매 두 번째 이터레이션마다 중간 인수 테스트를 수행한다. 중간 인수 테스트에 이어서 '보람'은 '고은', 영업, 마케팅 및 지원 팀과 함께 0.5일 동안의 백로그 미팅을 열어서 게임에 대한 새로운 아이디어가 있는지 확인하고 다음 미니 인수 테스트에서 그들이 보고 싶어하는 것을 알아낼 것이다. 그들의 요구 사항은 다음과 같다.

- 다음 중간 인수 테스트에 필요한 요구 사항
- 7월 릴리즈를 위한 요구 사항
- 7월 이후까지 보류할 요구 사항

5.3 릴리즈 계획

5.3.1 릴리즈 목표

릴리즈 계획을 시작할 때 목표 및 비전을 세워라. 고객과 제품 책임자는 고객의 비전 및 그것이 어떻게 프로젝트 목표에 부합할지 논하라. 릴리즈의 예시는 아래와 같다.

예제 22. 릴리즈 계획의 이유

- **이벤트 :** 특정한 날의 무역 박람회 또는 전시. '고은'의 회사는 게임 소프트웨어를 만들며 그들의 거대한 무역 박람회는 7월에 있다. '고은'은 '릴리'(Lily)프로젝트의 7월 새 버전 릴리즈 진척을 확인한다.
- **시즌:** 휴가철과 같은 시장 최다 성수기. '광조'는 여행사의 CEO이다. 그는 웹사이트가 여름 휴가철의 최다 구매 시기에 대비해 업데이트되는 것을 필요로 한다. 그는 '마이트래블(MyTravel)'프로젝트 릴리즈를 마케팅 전략과 조율시키기를 원한다.
- **규정과 법률 관련 :** 신규 혹은 변경된 판매세율 또는 규제 업데이트를 충족시킬 필요. '나영'은 재정 고문 그룹의 COO이다. 그들은 올 회계연도 말까지 새 규정을 준수할 필요가 있다. '로즈(Rose)' 프로젝트의 소프트웨어 업데이트 릴리즈는 반드시 규정이 발효되는 날과 부합해야 한다.
- **사용자 이용가능성 / 의향:** 사용자는 그들의 소프트웨어의 변경을 원하지 않을지도 모르며 인수 테스트를 할 수 없을지도 모릅니다. 이 점은 계약서에 제시돼 있을 것이다. '남선'은 COTS소프트웨어를 구축한 회사의 마케팅 관리자이다. 그는 'PIP'프로젝트의 고객이다. 그는 자신의 사용자가 업데이트를 일년에 두 번 이상 받아들이길 원하지 않는다는 것을 알고 있다. 그는 사용자와 합의된 지정 일자에 릴리즈 하기를 원한다.
- **비상시:** 사용자는 일을 지속할 수 없을지도 모르며 정상 릴리즈 시점 사이에 그들의 소프트웨어 변경을 급히 요구할지도 모른다. '덕수'의 업무 지원센터 운영자 팀은 그들이 사용하는 소프트웨어에 문제를 겪고 있다. 그 문제가 매우 심각해 그들은 고객의 문의사항을 기록 및 처리할 수 없을 정도이다. '덕수'는 'POP'팀이 이터레이션 기간에 일시적 변경을 유발할 소프트웨어 긴급 수정 버전을 릴리즈 하도록 요구한다.
- **에픽:** 한 이터레이션 이상이 소요될 때에는 '에픽' 사용자 스토리가 필요하며 사용자는 부분이 아닌 완전한 사용자 스토리를 받기 원한다. '동건'은 결혼 준비를 위해 여행자 웹사이트에서의 새로운 기능을 요구했다. '동건'은 이것이 한 사용자 스토리보다는 더 복잡하다는 것을 알고 있다. 이것은 '마이트래블(MyTravel)'프로젝트 동안 여러 이터레이션 구간에 걸쳐 구축할 에픽인 것이다.
- **비기능 테스트:** 성능, 보안, 신뢰성, 사용성, 회귀 또는 규정 테스트는 한 이터레이션 내에서 적합하지 않을 수 있다. '경모'는 'PIP'프로젝트의 성능을 테스트하고 있다. 성능 테스트는 모든 세 번째 이터레이션에서 수행되며 이는 릴리즈 계획에 반영돼야 한다.
- **기술적 제약:** 한 이터레이션 이상이 소요될 기술적 변경 요구가 있다. '무열'은 한 이터레이션 내에서 수행될 수 없는 '감마'(Gamma)프로젝트의 아키텍처에 대한 변경 작업을 수행한다.

5.3.2 릴리즈 내용에 대한 합의서

모두가 동의하는 합의서를 작성하는 것은 중요하다. 제품 책임자는 다음 릴리즈에 포함돼야 하는 것을 요약하며 모든 팀원은 성과가 무엇인지 명확하게 알게 된다.

사용자 스토리들은 그것들 사이의 식별된 중요성과 의존도 순으로 놓인다. 이것은 제품 백로그 관리 회의에서 처리될 수 있다.

가능한 것에 모두가 동의할 수 있도록 모두가 팀의 속도–즉 팀이 한 이터레이션 내에서 지속 가능한 속도로 얼마나 많이 작업할 수 있는지–를 이해하는 것은 중요하다.
각각의 사용자 스토리는 우선순위와 추정치를 가진다. 이터레이션에 배정하고 더 많은 정보를 알게 되면 상세한 이터레이션 계획이 변경될 것이다.

5.3.3 릴리즈 계획 문서화

제품 책임자는 동의된 사항을 기록하며 이를 모두가 볼 수 있는 곳에 전시한다. 팀 리더는 작성된 스토리들을 아래와 같은 릴리즈 상황판을 만들어 팀원들이 볼 수 있도록 벽면에 부착하거나 도구(Excel, Redmine 등)에 저장한다. 각각의 이터레이션에는 팀이 소화할 수 있는 양만큼의 스토리들을 포함시키며 이터레이션 초기에는 팀의 평균 속도보다 적은 양을 할당하고 이후 늘려간다. 마지막 이터레이션은 보통 통합테스트에 할당하거나 버퍼로 확보한다. 팀 리더는 첫 번째 릴리즈에 해당하는 릴리즈 번–업 차트를 작성하며 이것을 공개적으로 전시한다.

이터레이션 1	이터레이션 2	이터레이션 3	이터레이션 4	이터레이션 5	이터레이션 6
S □, S □ S □,	S □, S □ S □, S □	S □ , S □ S □ , S □ S □	S □ , S □ S □ , S □ S □ , S □	S □ , S □ S □ , S □ S □ , S □	통합테스트

5.3.4 모두에게 알리기

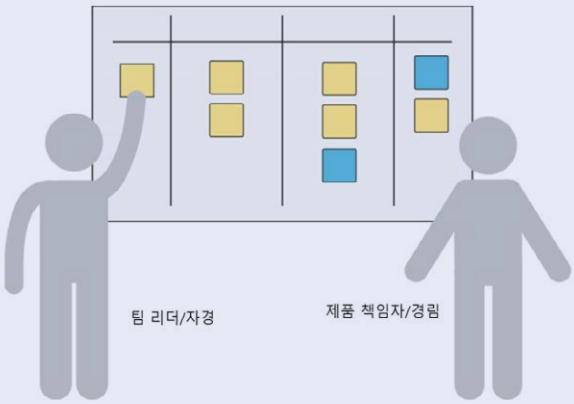
Tip 35. 제품 책임자와 바쁜 사용자

많은 사용자는 제품 책임자와 이야기하기엔 너무 바쁘다. 그들은 프로젝트에 참여할 뿐만 아니라 다른 해야 할 일들이 있다. 그래서 종종 제품 책임자들과 이야기할 시간이 거의 없다. 그들을 방문하거나 그들이 팀을 방문한다면 명확하고 간결하며 정직해야 함을 기억하라. 항상 보이는 작업 현황판과 번-업 차트는 모두가 현재의 프로젝트 상태를 볼 수 있도록 최신의 것이어야 한다. 또한 그들이 제품 책임자를 그들의 대리인 및 의사소통자로 임명했음을 기억하라. 만일 프로젝트에 어떤 문제가 있다면 항상 제품 책임자와 팀 리더에 의해 첫 번째로 논의돼야 한다.

예제 23. 제품 책임자와의 의사소통

▶ 팀 리더는 제품 책임자와 협업을 위해 번-업 차트를 사용한다.

'경림'은 'XYZ'프로젝트의 제품 책임자이지만, 그는 또한 사용자로서의 별개의 역할도 있으며, 또 다른 더 복잡한 프로젝트의 제품 책임자이다. 그는 팀이 위치해 있는 층을 매일 걸어서 지나갈 것이며 작업 현황판을 확인할 것이라는 점에 대해 팀 리더인 '자경'과 합의한다. '자경'은 작업 현황판을 모두가 걸어가며 보기에 분명하고 쉬운 곳에 걸어두며 최신 상태가 되도록 한다. 때때로 제품 책임자는 진척에 관한 차트 앞에 멈춰서지만 대부분은 빠르게 훑어보고 지나쳐 간다.



5.4 이터레이션 계획

5.4.1 파트1 : 이터레이션 백로그 선택

두 이터레이션 계획 회의(이터레이션 백로그 선택) 중 첫 번째에서 제품 책임자와 팀은 이터레이션 동안 무엇이 수행될지에 동의한다. 회의에서 아래와 같은 일을 수행한다.

- 어떤 사용자 스토리들이 요구되는지 확인하기
- 스토리들 간 의존성 확인하기
- 사용자 스토리에 배정될 스토리 포인트 추정하기
- 사용자 스토리들의 우선순위와 팀의 속도에 기반해 이터레이션 내에 완성될 수 있는 사용자 스토리들 선택하기

• 이터레이션 목표 이해 •

이터레이션 목표는 릴리즈 목표에 연관된 이터레이션 내에서 성취돼야 하는 것을 말한다.

예제 24. 이터레이션 목표 예시

▶ '마이트래블(MyTravel)'팀의 이터레이션 목표 수립

'마이트래블(MyTravel)'팀은 프런트 오피스 시스템 사용자를 위한 효율성을 개선하는 다음 이터레이션을 위한 이터레이션 목표를 갖고 있다.

1. 중요하고 빈번한 시스템 영역의 성능
2. 사용자 경로와 중요하고 빈번한 사용자 태스크의 인터페이스

• 사용자 스토리 사례 •

제품 책임자는 이터레이션 목표에 기여하는 사용자 스토리들의 목록을 가지고 있다. 이러한 사용자 스토리들은 다음을 고려해 완성돼야 한다.

- 이름과 기능 요구사항
- 비기능적 요구사항
- 담당자
- 우선 순위
- 인수 기준

이터레이션 계획 회의에서 팀은 다음을 확인함으로써 시작한다.

- 선택된 사용자 스토리들을 지원하는 데 필요한 기술적 스토리들
- 완성되기 위한 다른 사용자 스토리들에 대한 의존성

팀과 제품 책임자는 이터레이션 동안에 완료 정의가 조정이 필요한 지 논의할 것이다.

예제 25. 완료 정의 변경

▶ '마이트래블(MyTravel)'팀을 위한 완료 정의 변경

'마이트래블(MyTravel)'팀은 요구되는 성능 튜닝에 관해 논의한다. 그들은 이번 이터레이션에 성능 테스트가 필요하다는 것을 완료 정의에 추가하기로 결정했다.

• 이터레이션 기간 추정하기 •

이터레이션 내에 가용한 날들을 확인하기 위해 다음을 파악한다.

- 이터레이션 제로에서 동의한 인수 테스트, 계획, 회고를 위해 사용되는 이터레이션 기간과 시간에 대한 합의
- 이터레이션을 위한 알려진 부재: 훈련, 휴가, 알려진 병가, 회사 회의 등

예제 26. 이터레이션 기간은 얼마나 되는가?

▶ 이터레이션 기간은 얼마나 되는가?

'PIP'프로젝트에 관한 팀은 4 명이다. 이터레이션8을 위해 그들은 보통의 이터레이션 기간인 2주와 3일간의 다음 계획, 인수 테스트, 배포와 회고, 그리고 7일간의 개발 작업(설계, 코딩, 테스트)을 추정한다. 팀에서 한 명은 하루 휴가가 있다. 따라서 이 이터레이션 내에는 (7*4)−1 = 27일만큼의 가용한 날이 있다.

• 추정하기 – 이전 이터레이션에 기반한 속도 •

지난 3개의 이터레이션에서 실제로 완료했던 것에 기반해 속도를 추정한다. 아래에 프로젝트의 첫 다섯 이터레이션을 통해 속도를 추정하는 예제가 있다.

예제 27. 속도는 얼마인가?

▶ 속도는 얼마인가?

'PIP'팀은 이터레이션9의 끝에 있다. 지난 세 이터레이션에서 인도된 스토리 포인트를 리뷰 한다.

이터레이션	이터레이션 백로그 내 스토리 포인트	완료된 스토리 포인트
7	60	60
8	60	61
9	60	59

지난 세 이터레이션의 평균 속도는 60스토리 포인트이다.

• 추정하기 – 속도를 일시적으로 바꿀 요소들 •

이전 이터레이션에 기반한 속도는 팀이 약간의 이터레이션을 경험하고 일이 일정하게 진행될 때 잘 작동한다. 프로젝트 내 특정한 지점에서 팀의 속도는 다른 요소들에 영향을 받을 수 있다.

- 첫 세 이터레이션에서는 팀이 프로젝트에 익숙해지는 동안에 더 낮은 속도를 추정하는 것이 더 낫다.
- 팀이 아래와 같은 상황이면 더 낮은 속도가 될 것이다.
  - 리드미가 처음인 경우
  - 새로 설립된 팀일 경우
  - 새로운 비즈니스 영역일 경우
  - 기술 수준이 낮을 경우
- 아래와 같은 경우 속도가 낮아질 것이다.
  - 새로운 팀원이 프로젝트 도중에 참여할 때 다른 팀원이 새 멤버를 팀에 참여시키는 것을 돕느라 시간을 보내고 있다면 이런 경우에 "팀원 참여"를 이터레이션 백로그에 스토리로서 넣는 경우
  - 한 팀원이 프로젝트 동안 새로운 기술을 배울 필요가 있고, 이것이 팀의 소프트웨어 인도 일정을 변경시킬 때, 이런 경우에 "기술 훈련"을 이터레이션 백로그에 사용자 스토리로서 추가하는 경우

• 추정하기 – 이터레이션 백로그를 위한 스토리 포인트 •

전형적인 이터레이션에서 팀의 속도와 스토리들이 이터레이션 기간 내에서 얼마나 많이 수행될 수 있는가에 동의하며 최우선 순위 사용자 스토리들의 포인트 추정치를 사용한다. 이 이터레이션 기간에 집중할 수 있는 것에 대한 추정치를 조정하기 위해 위의 모든 요소들을 고려해야 한다.

이터레이션 끝에, 사용자에게 받아들여진 사용자 스토리로부터의 스토리 포인트를 계산할 수 있으며 이터레이션 속도를 알 수 있다. 부분적으로 완성된 스토리들은 스토리 포인트를 계산하지 않는다.

• 추정치 논의하기 •

모든 팀원은 추정에 참여한다. 팀원이라면 이터레이션 계획 동안 위에서 보았듯이 스토리 포인트 내에서 각 스토리의 크기를 제안할 것이다.

서로 다른 사람들이 한 스토리에 서로 다른 크기들을 제안한다는 점을 발견할 것이다.
그 이유는 다음과 같다.

- 각 사람이 완성돼야 할 작업에 대해 서로 다른 경험이 있다.
- 팀 동료들이 알지 못하는 리스크나 문제점을 알고 있다.
- 팀 동료들이 알지 못하는 빠른 방법을 알고 있다.
- 누가 그것을 행하는지에 따라 다르다.
- 추정 경험이 부족할 때 추정치가 정확하지 않다.
- 충분히 알지 못하기 때문에 추정하는 것이 불가능할 수도 있다.

추정치를 제공하기에 충분한 정보를 갖지 못한다면 질문을 하라. 하지만 이것이 추정일 뿐임을 기억하라. 틀릴 수도 있다고 예상하라.

모든 추정치와 추정의 이유 및 논의에 귀 기울이며, 팀 리더는 각 사용자 스토리에 사용될 스토리 포인트의 추정된 숫자를 정한다.

• 협상하기 •

팀의 속도와 계획된 부재 관련 정보를 사용하면서, 팀 리더는 스토리 포인트의 추정치가 다음 이터레이션에 수행될 수 있을 거라고 추정한다. 팀 리더라면, 팀의 현재 속도를 알 수 있도록 위에서 보았던 것과 같은 표를 작성할 필요가 있다.

후보 스토리들이 이터레이션 내에 맞는지를 팀원과 검토할 필요가 있다. 프로젝트 책임자와 함께 스토리들이 이터레이션 내에 맞지 않으면 제거될 수 있다는 것에 동의해야 한다. 또한 일련의 이터레이션에 걸쳐 완성될 수 있도록 스토리들을 추가하거나 더 작은 스토리들로 나눠지게 요구할 필요가 있다.

또한 한 이터레이션 내의 작업이 팀 내에서 갖고 있는 기술 역량과 균형을 이룰 수 있도록 확인할 필요가 있다. 누군가 새로운 기술을 배우길 원한다면 그들이 기술을 배울 추가적인 시간을 허용할 필요가 있다.
이터레이션이 전달할 것에 대해 제품 책임자와 협상해야 한다. 또한 스토리들이 그 이터레이션 내에 전달될 수 있음을 확실히 하기 위해 팀과 협상해야 한다.

일단 제품 책임자 및 팀과 동의했으면, 모두가 그 이터레이션 백로그를 전달하는 것에 전념하도록, 그리고 그 이터레이션 백로그가 그 이터레이션 동안 변하지 않을 것임에 동의하도록 요구한다.

Tip 36. 이터레이션 백로그가 많거나 적지 않게 하기 위해 협상하기

이터레이션 백로그가 완성되면 이를 보고 팀이 수행할 모든 것을 알게 된다. 너무 많지도 적지도 않게 하기 위해 충분한 고려를 해야 한다.

예제 28. 이터레이션 계획 협의

▶ 이터레이션 백로그가 팀이 수행할 수 있는 것보다 더 많은 작업을 담고 있을 때 반드시 제거돼야 한다.

'무영'은 'PIP'프로젝트의 제품 책임자이다. 그는 12개의 사용자 스토리가 이터레이션 내에 완성되기를 요구한다. 스토리 전달의 추정치는 전부 합해서 42스토리 포인트이다(우선순위1이 가장 높다).

S1 – (5 스토리 포인트) 우선순위 1
S2 – (8 스토리 포인트) 우선순위 1
S3 – (3 스토리 포인트) 우선순위 1
S4 – (1 스토리 포인트) 우선순위 2
S5 – (2 스토리 포인트) 우선순위 2
S6 – (8 스토리 포인트) 우선순위 2
S7 – (5 스토리 포인트) 우선순위 2
S8 – (8 스토리 포인트) 우선순위 2

S9 – (3 스토리 포인트) 우선순위 3
S10 – (2 스토리 포인트) 우선순위 3
S11 – (5 스토리 포인트) 우선순위 3
S12 – (5 스토리 포인트) 우선순위 3

'PIP'프로젝트의 팀은 4명이며, 그들은 각각 개발 및 테스트 작업을 수행한다. 그들의 현재 속도는 한 이터레이션 당 42스토리 포인트이지만, '재규'가 이터레이션 기간 동안 병가를 내서 팀은 한 사람이 부족하다(프로젝트의 이터레이션 10) – 그러므로 팀의 예상된 속도는 3 / 4 * 42 = 31.5 스토리 포인트로 다시 계산된다. '무영'은 우선순위 3인 스토리 2개(S11와 S12)를 다음 이터레이션으로 옮긴다. 팀은 총합해 32 스토리 포인트인 10개의 사용자 스토리가 있는 이터레이션 백로그에 전념한다. '재규'가 돌아올 때 그의 복귀를 고려해 속도를 다시 계산해야 한다.

5.4.2 이터레이션 계획 동안 스토리 포인트와 속도의 작업 예시

• 이터레이션 1 •

이터레이션 1을 시작할 때, 이터레이션 제로 논의로부터 이미 다음을 알 수 있다.

- 합의된 이터레이션 기간
- 개발의 합의된 가용 일 수
- 스토리 포인트의 정의
- 속도에 대한 추정

이것이 첫 번째 이터레이션이고 53 스토리 포인트를 추정했다. 아직 부정확한 추정치이다.
이터레이션 끝에 스토리 포인트는 아래와 같다.

이터레이션	이터레이션 백로그 내 스토리 포인트	완료된 스토리 포인트
1	53	42

17개 스토리, 53스토리 포인트를 이터레이션 백로그에 넣는 것에 동의했고, 그들 중 15개를 완료했지만, 두 개의 스토리(스토리 포인트 3, 8)는 이터레이션 말까지 완성되지 않았다. 그러므로 첫 이터레이션 속도는 42이다.

• 이터레이션 2 •

두 번째 이터레이션의 백로그 내용에 동의할 때, 새로운 속도를 사용할 수 있다. 두 번째 이터레이션의 말에 스토리 포인트는 아래와 같다.

이터레이션	이터레이션 백로그 내 스토리 포인트	완료된 스토리 포인트
1	53	42
2	48	48

두 번째 이터레이션 내에서 더 잘 할 수 있다고 생각했었으며 6 스토리 포인트까지 증가하면서 적게 잡은 속도의 추정치를 충족하는 데 성공적이었다는 것을 확인할 수 있다.

• 이터레이션 3 •

속도는 이제 45(첫 두 이터레이션의 평균)이지만, 그것보다 훨씬 더 잘 할 수 있다고 느끼므로, 세 번째 이터레이션에서 57 스토리 포인트까지 수행하기로 계획한다.

이터레이션	이터레이션 백로그 내 스토리 포인트	완료된 스토리 포인트
1	53	42
2	48	48
3	57	57

• 이터레이션 4와 5 •

세 번째 이터레이션에서 57 스토리 포인트까지 해냈다. 속도는 이제 49이다. 점차 이터레이션 속도는 전체 평균 속도가 아니라 마지막 세 이터레이션 평균으로 계산할 것이므로 초기 이터레이션으로 내려가지는 않는다.

5개의 이터레이션 후에 아래와 같이 스토리 포인트를 기록해 두었다.

이터레이션	이터레이션 백로그 내 스토리 포인트	완료된 스토리 포인트
1	53	42
2	48	48
3	57	57
4	60	58
5	60	59

[표2. 속도 계산]

이제 경험을 가지고서 초반보다 더 많은 스토리 포인트 값으로 스토리들을 완료하고 있음을 확인할 수 있으며 또한 추정을 더 잘 하게 됐다. 현재 속도는 58(지난 세 이터레이션의 평균)이다.

5.4.3 파트2: 팀 활동 계획

두 이터레이션 계획 회의(팀 활동 계획) 중 두 번째로, 팀은 방금 이터레이션 백로그 내에 포함시키기로 합의한 스토리들을 어떻게 전달할지를 정한다. 이 회의에서 다음을 수행한다.

- 누가 어떤 활동에 대해 책임질 지와 이터레이션 백로그에서 어떤 스토리들에 책임을 맡을지 정하기
- 스토리를 완료하는 데 필요한 전문가 도움 확인하기
- 스토리들 간의 우선순위와 의존도에 기반해 이터레이션 기간 동안의 작업 일정 정하기
- 관련 사용자를 인수 테스트에 참여하도록 초대하기

• 작업 현황판 •

팀과 제품 책임자와 팀 리더가 이터레이션 백로그에 동의하면 팀 리더는 모든 태스크를 작업 현황판에 기록한다.

Tip 37. 어떤 순서로 사용자 스토리들을 작업 현황판에 게시할 것인가?

처음에는 어떤 순서로 카드들을 작업 현황판에 올리는지는 중요하지 않으며, 일정관리 하는 동안에 어떤 순서로 카드들을 작업할지 토론하게 될 것이다. 처음으로 그것들을 작업 현황판에 놓을 때, 전부 이터레이션 백로그 첫 세로줄에 넣으면 된다.

'PIP'프로젝트의 이터레이션10의 작업 현황판 설정 예시

대부분의 팀원은 처음에 프로그래밍, 테스팅, 배포와 같은 특정 기술로 시작하지만 팀에서 이들은 곧 이터레이션 동안의 다양한 활동에 도움이 되기 위해서는 다양한 기술이 필요하다는 것을 알게 된다. 일반적으로 팀원은 여전히 한 영역에 특히 숙련된 것으로 간주되겠지만 이터레이션 전체 기간 동안 팀에 기여할 수 있는 새로운 기술도 습득할 것이다.

예제 29. 이터레이션 작업 현황판

PIP이터레이션10:
10개의 스토리. 예상 속도= 32 스토리 포인트

재규 얼른 회복하기를- 그립다!

이터레이션 백로그	개발	스토리 테스트	인수 테스트	배포
S2 ☐ S6 ☐ S8 ☐ S1 ☐ S7 ☐ S3 ☐ S9 ☐ S4 ☐ S5 ☐ S10 ☐				

• 추가적 전문가 기술과 자원 식별하기 •

팀은 스스로 완료할 수 있는지 또는 전문가 도움이나 자원이 필요한지 식별해야 한다. 팀 리더는 어떤 추가적인 사람들과 자원들이 필요한지에 대한 목록을 작성한다.

예제 30. 추가적 자원의 필요성 평가

예시: '마이트래블(MyTravel)'팀은 성능 테스트를 위해 전문가 조언이 필요하다. 전문가가 성능 튜닝을 위해 설계 변경을 검토해주기를 원한다.

• 일정관리 •

일정관리는 팀이 얼마나 빨리 작업 하는지- 어떻게 작업 현황판을 가로질러 이동할지, 어떤 순서로, 그리고 누가 무엇을 하는지의 순서를 정하도록 돕는다. 종종, 특히 작은 팀에서, 이러한 일정관리는 비공식적으로 행해지며 전혀 문서화되지 않는다. 일정관리는 작업 현황판에 추가될 수 있다.

'PIP'프로젝트의 이터레이션10의 일정관리 후 작업 현황판의 예시

'나영'은 말한다. "이 작은 스토리들을 가능한 빨리 '스토리 테스트'로 이동시킬 것이며, 설계가 끝난 후 더 큰 스토리들의 개발을 도울 것이다." 테스트를 위해 어떤 특정한 순서로 해야 하나요?" '하늘'은 이에 대해 생각한다. "새로운 기능에 앞서 모든 결함 수정을 먼저 할까요? 그리고 나서 다른 변경이 있기 전에 회귀 테스트 통과를 확실히 할 수 있어요." "물론이죠!" 그 팀은 가장 큰 세 스토리들(S2, S6, S8)을 초기 설계를 위해 표시하는 반면 결함 있는 스토리들(S4, S5, S10)을 첫 번째 스토리들로 표시한다.

예제 31. 일정관리 후 이터레이션 작업 현황판

PIP이터레이션10:
10 개의 스토리. 예상 속도= 32스토리 포인트.

재규 얼른 회복하기를 - 그립다!

이터레이션 백로그	개발	스토리 테스트	인수 테스트	배포
S4 ☐ 수정 우선 S5 ☐ 수정 우선 S10 ☐ 수정 우선 S2 ☐ 설계 우선 S6 ☐ 설계 우선 S8 ☐ 설계 우선 S1 ☐ S3 ☐ S7 ☐ S9 ☐				

• 인수 테스트에의 초대 •

작업 현황판이 준비 되었으므로, 팀은 인수 테스트에 어떤 사용자가 필요할지에 대해 생각한다. 팀 리더는 날짜와 그들이 테스트하도록 사용자 스토리 및 필요한 다른 정보를 알려주며 관련 사용자를 초대한다.

'PIP' 팀 - 인수 테스트에의 초대

팀 리더인 '훈'은 제품 책임자인 '무영'에게 연락한다. 그는 그들에게 어떤 사용자 스토리들이 테스트될지, 언제 인수 테스트가 수행될지 말해준다. 그는 말한다. " 약간의 수정과 새로운 기능들을 포함해서 준비된 10개의 사용자 스토리가 있다."

5.5 일일 스탠드 업 미팅

일일 스탠드 업 미팅은 모든 근무일 마다, 가급적이면 하루 업무 시작 시에 개최된다. 팀원 모두는 이에 참여한다. 팀이 진행상황, 장애물, 다음 할 일을 보고할 기회이다.

5.5.1 회의 준비

• 팀 리더 준비 •

리더는 회의를 중재하고 촉진하는 역할을 한다. 회의가 원활하게 진행되도록 다음을 준비한다.

- 작업 현황판이 놓인 팀 내 공간
- 타이머 또는 시계
- 만일 회의에 참석할 수 없으면 적절한 사람 선정

• 팀원 준비 – 전문가들 포함 •

기술적 및 팀 관점에서 공유해야 할 사항을 준비한다.

- 한 일과 하려고 하는 일 공유
- 장애물 공유

5.5.2 회의 진행

• 회의 권장사항 •

- 정시에 시작하기
- 정시에 마치기
- 15분으로 회의 마치기
- 한번에 한 사람씩만 발언하기
- 주의 깊게 듣기
- 말하기 전에 생각하기
- 분명히, 간결하게, 긍정적으로 의견 말하기
- 토론에서 모든 아이디어들 수용하기
- 모두는 회의의 결정 존중하기
- 회의는 토론을 위한 안전한 장소일 것

• 회의 시작 •

회의를 중재하고 있다면

- 모두를 환영하라
- 필요하다면, 모두에게 회의 권장사항과 엄격한 시간 엄수를 상기시키라

예제 32. 일일 스탠드 업 미팅 정시로 유지하기

▶ 15분으로 준수하기

'마이샵'(MyShop) 팀은 팀 리더인 '미란'을 포함해 5명으로 구성돼 있다. 제품 책임자는 '숙자'이다.

일일 스탠드 업 미팅은 길어야 15분이 걸리며, 때때로 그보다 적다. 진행상황을 보고하는 5명이 있다.

개개인 들에게는 그들의 진행상황, 장애물, 그날의 계획을 발표하는 데 길어야 3분이 주어진다. '미란'은 팀 리더이며 회의를 중재한다. 그녀는 '성한'에게 그의 보고에 대해 질문함으로써 시작한다.

'성한'의 요약: "저는 웹 쇼핑 리턴 사용자 스토리 테스트를 준비 했다. 저는 장애물이 없다. 오늘 저는 웹 쇼핑"나의 세부사항 변경"사용자 스토리의 설계 작업을 할 계획이다."

'미란'은 작업 현황판이 '성한'의 보고를 반영하는지 확인하고 '성한'에게 감사하며 '오성' 차례로 넘어간다. '오성'은 어제 힘든 날을 보냈다! "오, 어제는 참으로 안 좋은 날이었다! 여러분도 아시다시피, 저는 '숙자'와 "자동 이체를 위한 은행 세부사항들 설정"사용자 스토리가 수행되기를 말하고 싶었지만, 그녀와 연락할 수 없었다. 제가 그녀와 이야기하고 싶어할 때면 그녀는 결코 없다! 어쨌든, 그래서 제가 할 수 있는 한 뭔가를 시도하며, 그리고 나서 또 여러분은 믿지 않겠지만"

'미란'은 그를 멈춘다! "오성씨 간결하게 해주세요! 방해물이 있네요 – 즉 '숙자'와 이야기할 필요가 있군요. 이제 진행상황과 계획들 – 1분 안에 말해주세요! 그리고 나서, 당신과 저는 후에 이야기하고 상황을 공유해요."
'오성'은 끄덕인다. "죄송합니다. 저는 "웹 쇼핑 회귀" 사용자 스토리 테스트를 어제 마쳤고, 오늘은 "나의 세부사항 변경"스토리 테스트를 시작할 것이며 3일이 걸릴 예정입니다."

'미란'은 "감사합니다."라고 말하며 다음 사람으로 넘어간다. 그녀의 훌륭한 중재를 통해 일일 스탠드 업 미팅은 여전히 정시에 끝난다.

• 회의 마치기 •

모두가 무엇을 해야할 지 이해했는가와 후속 조치(장애물의) 목록을 확실히 하고, 모두에게 감사하며 회의를 마쳐라.

5.5.3 회의 후

회의 후 새로운 방안들이 이터레이션 작업 현황판에 또는 제품 백로그를 위한 스토리 카드에 올려질 것이다. 팀 리더는 필요 시 장애물을 다른 팀에 보낸다.

예제 33. 일일 스탠드 업 미팅 후 장애물 처리하기

▶ 장애물 다루기

'마이샵'(MyShop) 팀 리더인 '미란'은 팀원인 '오성'이 제품 책임자인 '숙자'와 만나도록 도울 필요가 있다. 일일 스탠드 업 미팅 후에 '미란'은 '오성'을 몇 분간 한쪽으로 데리고 간다. '오성'은 '숙자'와의 좌절감을 설명하며 '미란'은 주의 깊게 듣는다. "이제 기분이 좀 좋아졌나요? '숙자'와의 문제를 해결합시다." 그녀는 '숙자'를 보러 가고, '숙자' 또한 좌절감을 느끼고 있음을 알게 된다. – 그녀는 '마이트래블'(MyTravel)이라는 또 다른 프로젝트에 관해 작업해주도록 동시에 요구 받고 있었던 것이다! 그녀는 더 많은 시간을 '마이샵'(MyShop) 팀과 보내고 싶어하지만, 두 개의 프로젝트로 인해 시간 배정이 어렵다. '미란'과 '숙자'와 '마이트래블'(MyTravel) 프로젝트의 팀 리더인 '요한'은 어떻게 이 문제를 해결할 수 있을지 의논한다. 그들은 '숙자'가 각 팀과 보낼 수 있는 시간에 합의한다. '숙자'는 '오성'을 만나러 오며, 그들은 "나의 은행 세부사항들 설정" 사용자 스토리에 대한 그의 질문들을 해결한다. '미란'과 '숙자'는 함께 차를 마신다. "이 장애물이 해결돼 좋군요."

5.5.4 진척회의 대신에 일일 스탠드 업 미팅 하기

일일 스탠드 업 미팅은 진척회의 대신 사용될 수 있다. 팀의 진척회의가 여전히 고객 또는 제품 책임자에 의해서 관리 중이라면, 팀 리더는 반드시 그들과 함께 어떤 정보를 원하는지 의논해야 하고, 그들의 관심사를 이해하려고 노력해야 하며, 필요하다면 그들에게 코칭을 제안해야 한다.

예제 34. 진척회의를 대체하는 일일 스탠드 업 미팅

▶ 공식 진척회의를 대체하는 일일 스탠드 업 미팅

'미란'과 '요한'은 함께 점심식사를 한다. '요한'은 이 회사에서 수년간 근무해오고 있다. '미란'은 최근에 합류했으며 '요한'은 그녀를 지도해주고 있다. '요한'이 말한다. "당신도 아시다시피, 때때로 사람들의 장애물을 해결하며, 사람들이 서로간에 소통할 수 있도록 하는 팀 리더가 된다는 것은 힘들게 느껴질 수 있다. 하지만 긴 회의를 했었던 이전을 떠올릴 때면!" '미란'이 끄덕인다. "저는 당신에게 묻고 싶었어요. 정말로, 진척회의를 하지 않나요? 그럼 어떻게 고객이 진척 상황을 알 수 있나요?" '요한'은 웃는다. "맞아요. 지루하고 길고 시간낭비인 진척회의를 하지 않아요. 고객은 작업 현황판과 번-업 차트를 볼 수 있고, 한눈에 진척 상황을 알게 됩니다. 사용자는 동작하는 소프트웨어를 전달받고 있어서 진척이 일어나고 있음을 알게 됩니다. 제품 책임자와 팀 리더는 서로 항상 대화를 합니다. 그리고 매일 빠르게 장애물을 처리하며 그것들을 일일 스탠드 업 미팅에서 보고하고, 이후 곧바로 수정하므로 진척이 훨씬 더 원활합니다."

5.6 회고 미팅

회고 미팅은 각 이터레이션 말에 한다. 팀에서 잘 진행된 점, 잘 진행되지 못한 점, 그리고 팀의 개선을 위한 아이디어에 대해 생각해볼 기회이다.

- 가치 있는 소프트웨어 전달
- 지속 가능하게 작업
- 팀원 서로와 사용자와 의사소통 잘 하기
- 기술 향상시키기
- 효과적으로 일하기

5.6.1 회의 준비하기

• 모든 참석자들(제품 책임자, 팀 리더, 팀원 그리고 전문가들) •

이터레이션에 대해 무엇을 보고하고 제안할지 생각할 필요가 있다.

- 잘 진행되었던 것과 더 나아질 수 있는 것 대한 의견
- 프랙티스 개선에 대한 아이디어

• 팀 리더 준비 •

팀의 리더는 회의를 중재하며 회의가 원활하게 진행되도록 준비한다.

- 필요 시 화이트보드, 충분한 좌석, 테이블이 갖춰진 공간 예약하기
- 다과
- 타이머 또는 시계
- 제품 책임자 초대 및 팀에게 참석 상기시키기
- 다른 초대들은 필요 시에만 하기 – 연관됐던 또는 개선 아이디어를 가지고 있을지도 모르는 전문가들
- 개선 및 안건을 위한 이전 액션 아이템
- 팀 리더가 회의에 참석할 수 없다면 적절한 사람을 선정하라

• 제품 책임자 준비 •

제품 책임자는 회의에 참여한다. 노트와 아이디어를 준비하며 고객과 사용자를 대표한다. 회고 미팅 전에 사용자에게 피드백을 받고 비즈니스 관점에서의 의견을 가지고 회의에 참여한다.

- 팀 리더가 잘 진행되었던 것과 더 나아질 수 있는 것 대한 의견
- 프랙티스 개선에 대한 회의 진행

• 회의 권장사항들 •

- 정시에 시작하기
- 정시에 마치기
- 중간에 15분 휴식을 갖춘 반나절로 회의 시간 제한하기
- 한 번에 한 사람만 발언하기
- 주의 깊게 듣기
- 말하기 전에 생각하기
- 명확히, 간결하게, 긍정적으로 의견 말하기
- 모든 아이디어는 토론에 허용
- 회의의 결정 존중하기
- 안건을 자유롭게 제기하기

• 회의 순조롭게 진행하기 •

회의를 중재하는 사람은

- 환영하기
- 회의 권장사항 상기시키기
- 마침 시각과 휴식 시간 알리기
- 회의 시간 측정을 위한 타이머 사용하기
- 시간을 확인할 수 있도록 벽에 시계 놓기
- 시간을 낭비하고 있거나 너무 길게 말면 조정하기

• 회의 단계들 •

■ 시작 (10분)

회의 진행자는

- 계획된 속도 vs. 이터레이션에서 달성한 실제 속도 보고 (2분)
- 성공적인 결과물 보고(3분)

■ 아이디어 모으기 (최대 60분)

회의 진행자는 참여자가 각 라운드의 의견을 말하도록 기회를 주고, 화이트보드나 플립 차트에 있는 중요 사항을 보고한다.

- 1 라운드: 참여자는 이터레이션에서 잘 진행된 것을 한 가지씩 말한다.
- 2 라운드: 참여자는 이터레이션에서 잘 진행되지 못한 것을 한 가지씩 말한다.
- 3 라운드: 참여자는 경험을 통해 개선을 위한 아이디어를 한 가지씩 말한다.

회의 참여자는 의견을 빠르고 간결하게 말한다.

■ 명확화 (20분)

회의 진행자는 안건에 대한 질문이 있는지 확인하고 참여자는 필요시 간단한 토론을 할 수 있다.

■ 휴식 시간 (15분)

개선 아이디어에 대한 투표 (75분)

- 개선 아이디어의 우선순위에 투표. 참여자는 5표를 가지며 가장 우선순위가 높은 아이디어에 점을 찍는 "점 투표" 방법을 사용한다.
- 참여자는 5개의 우선순위가 높은 아이디어에 대한 공수를 추정한다.
- 다음 이터레이션 동안 즉시 수행될 수 있는 아이디어를 식별한다.
- 개선 아이디어를 팀원이 수행하도록 적절히 배정한다.
- 개선을 위해 어떤 조치들이 필요한지 확인한다.

■ 회의 마치기 (15분)

회의 진행자는 참여자가 결과에 동의하는지 확인하고 모두에게 감사하며 마친다.

5.7 개발

5.7.1 간결한 설계

간결한 설계: 미래에 유용할 것이 아닌 지금 필요한 것에 기반한 설계. 실용적이어야 함.

더 나은 접근법을 발견하지 않는 한, 간결한 설계를 프로젝트에 계속 사용하라. 팀이 설계를 가능한 간결하게 유지하는 것에 집중하는 것은 중요하다. 이는 이터레이션 내에 전달하는 것을 더 쉽게 해 주고, 소프트웨어를 미래 이터레이션에서 반복과 증분 동안에 변경시키고 유지하는 것을 더 쉽게 해 준다.

설계의 두 가지 측면에 대해 생각할 필요가 있다.

- **기술적 구조, 인프라, 설계**
  - 사용자에게 분명하고 사용자의 소프트웨어 사용을 지원하는 것
- **인터페이스, 메시지, 도움, 프롬프트, 명령어, 장치, 화면 크기, 플랫폼, 오디오, 시각, 설치 프로세스, 보안, 실행, 사용성, 신뢰성 등을 포함하고 있는 UX 설계**
  - 사용자가 보고 듣고 느끼거나 생각하는 것에 대한 모든 것

이터레이션 제로에서의 아키텍처와 설계는 프로젝트에서 조기에 수행할 만큼 간결할 필요가 있으며, 프로젝트 진행 동안 팀과 사용자를 지원할 만큼 충분히 견고할 필요가 있다. 빠르게 전달하는 것과 지속적이고 유지 가능한 설계를 전달하는 것 사이에서 적절한 선택이 필요하다. 너무 간단한 설계를 고른다면, 사용자에게 가치를 전달하는 대신에 설계를 재조작하는 것에 이터레이션을 사용하고 있을 것이다. 설계를 할 때, 아래 사항을 고려해라.

- **인수 기준에서의 비기능적 속성**
  - 실행, 보안, 신뢰성
  - 유지보수성
  - 이식성
  - 호환성
  - 사용성
- **가능한 설계를 간단하게 지속하면서 리팩토링에 대한 필요성 감소**

설계에 사용할 접근법들은 다음과 같다.

- **UML**
- **사용자 중심 설계**
- **기능 주도 개발(FDD)**

기능 주도 개발에 대한 내용은 5.11.1 기능 주도 개발 방법론을 참조하라.

5.7.2 코딩과 단위 테스트

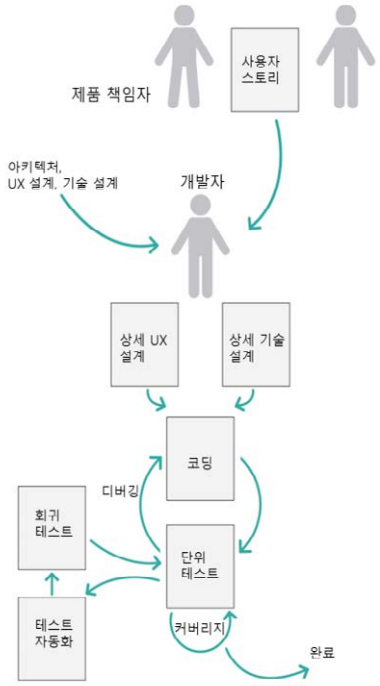
5.7.2.1 코딩 우선 프로그래밍

Tip 38. 코딩 우선이란 무엇인가?

코딩 우선 프로그래밍: 프로그래밍과 단위 테스트에 대한 코딩 우선 접근법은 코드를 먼저 개발하고 코드가 잘 동작하는지를 확인하기 위해 단위 테스트를 수행한다.

코딩 우선 개발 프로세스는 아래와 같다.

1. 사용자 스토리, 아키텍처, UX설계 리뷰
2. 사용자에게 추가적인 정보 확인
3. 스토리 기술적 설계 및 인터페이스 설계
4. 스토리와 설계를 충족시키는 코딩
5. 네거티브, 에러 추정을 포함한 테스트 설계
6. 가급적 자동화해 테스트 수행
7. 코드 커버리지 측정
8. 커버리지가 만족될 때 까지 수행



〈 도표 5. 코딩 우선 프로그래밍 〉

5.7.2.2 단위 테스트 – 도구 활용 가이드 참조

단위 테스트는 개발자가 코딩한 프로그램을 단위(모듈) 기준으로 수행하는 테스트이다. 단위 테스트에 대한 자동화 프레임워크가 많이 있으며 자세한 내용은 도구 활용 가이드를 참조하라.

5.8 테스팅

5.8.1 테스트 유형

프로젝트에서는 몇 가지 테스트 유형이 발생한다. 그것들은 다음과 같다

- 개발자가 수행하는 단위 테스트 (5.7.2 코딩과 단위 테스트 참조)
- 팀원이 수행하는 스토리 테스트
- 성능, 보안, 신뢰성, 사용성, 접근성과 같은 비기능 테스트
- 사용자와 제품 책임자가 수행하는 인수 테스트
- 회귀 테스트

이 장에서는 스토리 테스트, 비기능 테스트, 회귀 테스트를 설명한다.

5.8.2 테스트 대상 결정

테스트 준비는 간단하게 하라. 때때로 짝을 지어 일하는 것이 유용할 때도 있다. 한 사람은 소프트웨어를 운영하고 다른 한 사람은 제안하거나 코멘트와 노트를 작성한다. 유용한 접근법은 다음과 같다.

- **테스트에서의 중점 사항을 식별한다. – 이 스토리에서 중요한 것은 무엇인가?**
- **각 단계에서 무엇이 작동되고, 작동되지 않는지를 확인하면서 탐색적 테스트를 수행한다.**
  a. 스토리를 통해 기본 흐름을 수행한다. – 이것이 잘 작동하는가?
  b. 사용자가 소프트웨어에서 사용할 대안적인 방법에 대해 생각한다. – 이것이 잘 작동하는가?
  c. 리스크 수준이 높다면, 취약성이 있을 수 있는 곳에 대해 더 생각한다. 예를 들면, 경계 값은 입력 또는 출력 오류를 유발할 가능성이 높다. 테스트를 잘 설계하기 위해 테스트 설계 기법을 활용한다.
- **스토리에 대해 회귀 테스트 자동화를 원하는지 생각하고, 만일 원한다면, 잘 설계된 테스트 세트를 제공하도록 테스트 설계 기법을 활용한다.**

리드미에서 권장하는 테스트 설계 기법은 아래와 같다.

- 경계값 분석
- 동등 분할
- 상태 전이 테스팅
- 결정 테이블 분석
- 페어와이즈

이 설계 기법 중 일부는 도구 지원이 가능하다

5.8.3 회귀 테스트– 도구 활용 가이드 참조

Tip 39. 회귀 테스트

이터레이션 내의 변경이 기존 작업에 영향을 미치지 않음을 보장하기 위한 테스트. 요구사항 변경은 기존 소프트웨어 또는 인프라를 변경시킨다. 회귀 테스트는 다음 상황에서 필요하다.
– 단위 테스트의 부분으로서, 빌드마다 테스트 세트가 자동으로 수행될 때
– 스토리 테스트의 부분으로서, 가장 중요한 스토리 테스트 세트를 확인하고 자동화할 때
– 인수 테스트의 부분으로서, 가장 중요한 사용자 태스크를 확인하고 회귀 테스트를 보장할 때
회귀 테스트 세트는 매우 빨리 증가하므로 회귀 테스트 사용자 스토리의 작업량을 관리하는 방법을 권장한다.
– 리스크가 높은 사용자 스토리 테스트를 자동화하고 모든 이터레이션에서 자동화 수행하기
– 회귀 테스트를 더 일찍 하기 위해 단위 회귀 테스트 수행하기
– 리스크 기반 회귀 테스트 목록을 만들고 수행할 하위 목록 선택하기

회귀 테스트는 이터레이션에서 일어난 변경이나 결함 수정이 기존 소프트웨어에 영향을 미치지 않음을 보장하기 위해 필수적이며 모든 이터레이션 동안 관리돼야 한다.

어떤 스토리가 완료되기 위해서는, 그 스토리에 대한 회귀 테스트가 추가된다. 따라서 회귀 테스트 세트들은 점점 커지며, 가장 중요한 테스트 세트가 다음 이터레이션에서 수행되도록 우선순위가 매겨질 필요가 있다.
회귀 테스트는 반복적이며 수동으로 수행하기 어렵다. 가능하면 회귀 테스트를 자동화하라.

스토리 테스터나 인수 테스트를 돕는 팀원은 다음을 수행한다.

■ **결함이 수정된 후, 가장 중요한 스토리 및 인수 테스트를 식별하고 자동화 한다.**
- 모든 중요한 스토리 테스트들을 전부 자동화 하기에는 충분한 시간이 없다.
- 스크립트를 만들기 위해 테스트 설계 기법을 사용한다면, 설계 기법은 자동화뿐만 아니라, 자동화가 완료되기 전에 수동 회귀 테스트에도 사용될 수 있다.

■ **개발자들이 단위 테스트를 자동화하고 있다면, 스토리에 대한 회귀테스트도 단위 회귀 테스트 세트에 자동화하는 것이 최선이다.**
- 개발자들은 스토리 테스트 전에 회귀들을 식별하고 수정함으로써 시간과 노력을 절약할 수 있다.

5.8.4 비공식 결함 관리

▶ 비공식 결함 관리

스토리 테스트와 인수 테스트 동안, 이터레이션 동안 수정돼야 하는 결함들을 발견할 것이다. 이슈가 현재의 이터레이션 내에서 수정될 수 있다면 공식적으로 문서화할 필요가 없다. 만약 현재의 이터레이션 내에서 수정될 수 없다면, 이것은 제품 백로그에 추가하라.

결함 관리는 의사소통에 대한 것이다. 스토리 테스트 또는 인수 테스트 중에 결함을 발견하면, 테스트 세션 노트에 결함을 기록할 필요가 있다. 개발자에게는 결함에 관한 정보가 필요하다.

개발자에게 결함에 대해 설명하고 이터레이션 내에서 수정될지에 대해 개발자와 필요하면 팀 리더와 제품 책임자와 협의하라.

결함이 이터레이션 내에서 수정되면, 작업 현황판을 사용하라. 스토리 카드는 작업 현황판의 개발 열(column) 뒤로 보냄으로서 개발자는 결함을 수정해야 함을 알게 된다.

결함이 이터레이션 내에서 수정되지 않을 예정이면, 새로운 스토리 카드를 작성하고 제품 백로그에 추가하라.

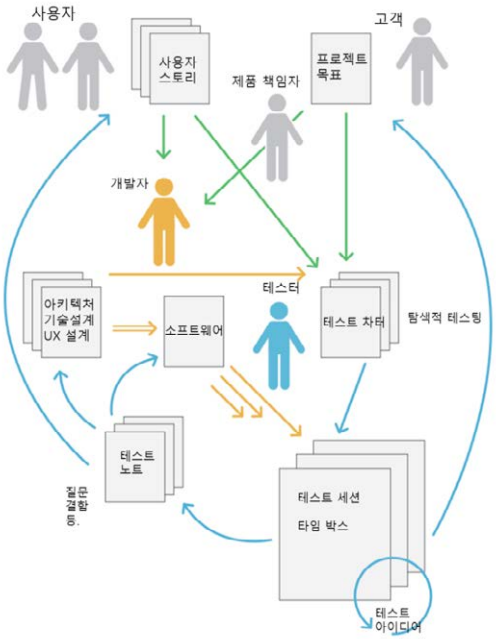
5.8.5 탐색적 테스트

▶ 탐색적 테스트

학습과 동시에 테스트 설계, 수행을 함. 테스트 결과에 따라 다음 테스트 영역이 탐색된다. 탐색적 테스트 동안에 테스트 과정, 결과, 결론, 잠재적인 결함 등을 충분히 기록한다.

스토리 테스트의 접근법으로 탐색적 테스트를 권장한다. 탐색적 테스트는 훈련된 접근법이며, 신중하고 효율적으로 테스트하게 한다.

이 그림은 이터레이션 내에서 한 개의 사용자 스토리를 위한 탐색적 테스트 프로세스를 보여준다.



〈 도표 6. 이터레이션 내의 탐색적 테스트 프로세스 〉

제품 책임자는 사용자 스토리를 제공하며, 고객은 프로젝트 목표를 수립한다. 개발자는 소프트웨어를 설계하고 개발한다. 그리고 나서 테스터는 탐색적 테스트를 수행한다.

탐색적 테스터는 다음을 수행한다.

■ **사용자 스토리, 사용자로부터의 정보, 프로젝트 목표를 분석해 테스트 차터를 작성한다.**
- 테스트 차터는 개략적으로 테스트 목표, 타임 박싱(time boxing), 리스크를 기술한다.

■ **탐색적 테스팅을 아래와 같이 수행한다. 보통 90분의 타임 박싱이다.**
- 테스트 세션(환경, 도구 등) 구성
- 테스트 차터, 사용자 스토리, 사용자로부터의 정보에 기반해 테스트 설계하고 수행
- 테스트 설계 기법 적절히 활용
- 개발자에게 충분한 정보를 제공하기 위한 결함 탐색
- 과정과 결과 노트하기

■ **테스트를 종료하고 세션 노트를 활용해 회고하기**
- 동료 테스터와 다음에 수행할 것에 대한 논의
- 개발자와 제품 책임자에게 질문을 하고, 결과와 결함을 보고

■ **위 프로세스를 반복**

테스트 세션에서, 중요한 것은 테스트 목표, 질문, 리스크, 남아있는 시간에 기반해 다음에 수행할 가장 중요한 테스트를 선택하는 것이다.

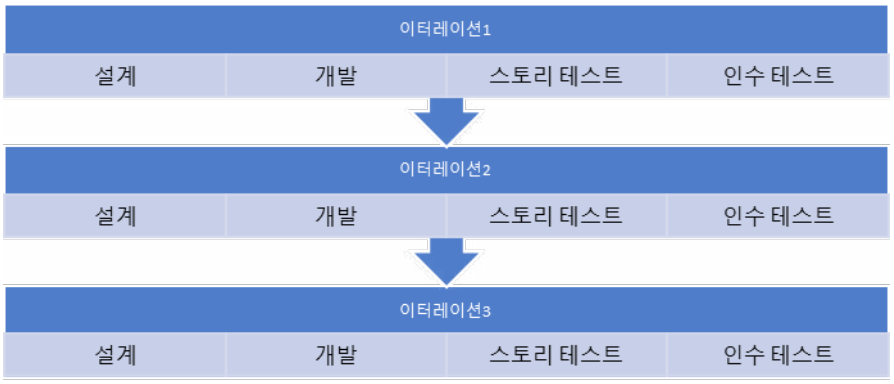
5.8.6 이터레이션에서 테스트 통합하기

▶ 테스트 통합

이상적인 이터레이션에서 모든 테스트는 팀에 의해 이터레이션 내에서 수행된다. – 개발자와 테스터가 서로를 동등하게 여기고 소프트웨어 품질에 대해 공동 책임을 지는 통합된 접근법

에서는 "테스트 통합"을 권장하는데 이는 개발과 스토리 테스트가 동일한 이터레이션 내에서 일어난다는 것을 의미한다.

팀은 함께 있으며, 개발자와 테스터는 같은 공간에서 작업한다. 함께 일할 수 있고, 의사소통 할 수 있고, 함께 짝을 지어 테스트할 수 있다. 함께 있는 테스터와 개발자는 소프트웨어에 대한 정보를 분명하고 시기 적절하게 알 수 있다.



〈 도표 7. 이터레이션에서의 테스트 통합 〉

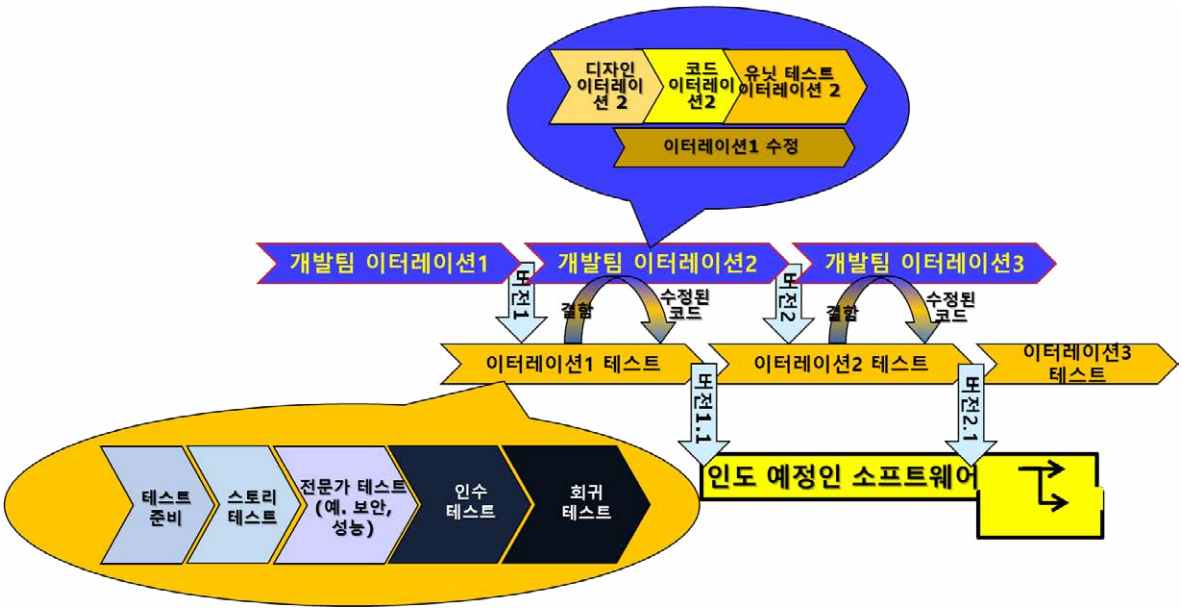
5.8.7 테스트 통합의 대안

때때로 테스트 통합이 어려울 수 도 있다. 예를 들어, 성능 테스트 환경이 가능하지 않을 수 있다. 대안이 있지만, 그 대안이 작업을 증가시키고 의사소통을 감소시키며 빌드에 오류 가능성을 증가시키므로 권장되지는 않는다.

• 병행 테스트 '이터레이션' •

병행 테스트 이터레이션은 권장되지는 않지만 필요할 수도 있다.
병행 테스트 이터레이션에서, 테스터는 아래 그림과 같이 동일한 이터레이션이지만 개발자 팀의 약간 뒤에서 작업하는 분리된 테스트 팀으로서 일한다. 테스트 팀은 먼저 준비하고 마지막 개발 이터레이션으로부터의 결과를 테스트한다. 테스트한 결함 보고서를 개발 팀에 보내며 개발 팀은 다음 이터레이션에 추가한다.

병렬 테스트 이터레이션으로 작업하는 개발자는 두 태스크가 있다. 이터레이션 백로그를 완료하기 위해 사용자 스토리를 작업하고 있을 것이며, 이전 이터레이션으로부터의 디버깅과 결함 수정을 하고 있을 것이다. 형상관리가 더 힘들 것이다. 그리고 사용자 스토리를 전환하면서 일을 하기 때문에 더 힘들 것이다.

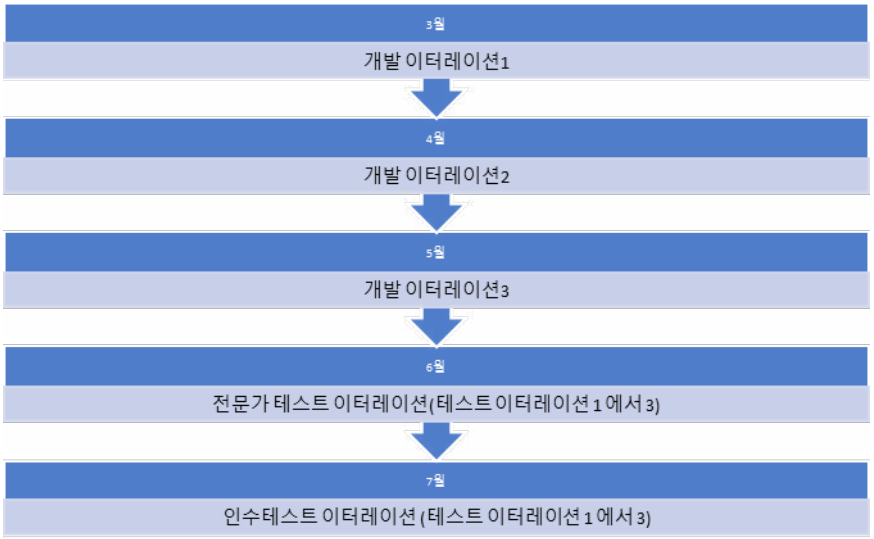


〈 도표 8. 병렬 테스트 이터레이션이 동반된 3개월 프로젝트 〉

• 가끔 수행하는 테스트 이터레이션 •

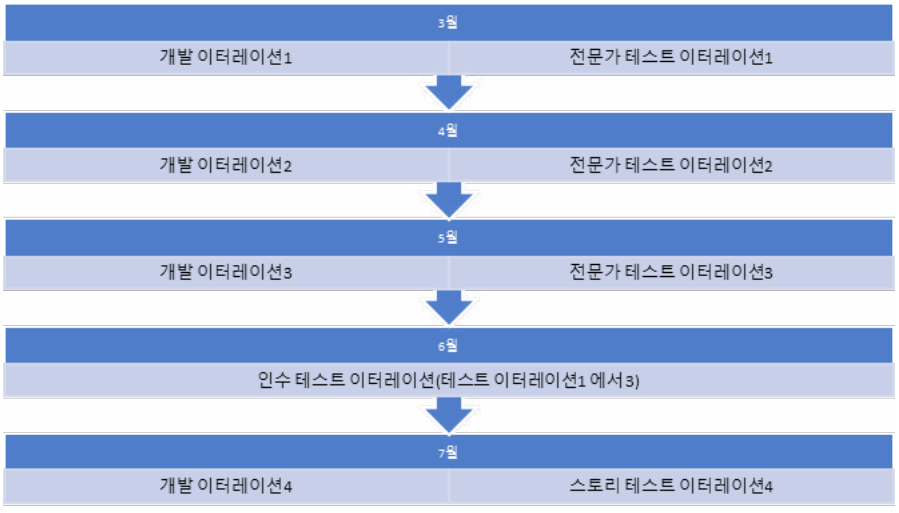
권장되지는 않지만, 사용자가 가끔씩 인수 테스트를 할 수 있거나 테스트 환경이 가끔씩만 가용 하다면 필요할 수도 있다.

병렬 테스트 이터레이션 대안은 아래 그림처럼 '정상적인' 개발 이터레이션 후의 결과물에 대한 '테스트 이터레이션'를 수행하는 것이다.



〈 도표 9. 5개월 프로젝트: 전문가 및 인수 테스트 이터레이션 포함 〉

때때로 전문가 테스트는 이터레이션 내에 위치하며 인수 테스트는 가끔 수행하는 테스트 이터레이션에 위치한다.



〈 도표 10. 5개월 프로젝트: 인수 테스트 이터레이션 포함 〉

5.9 지속적 통합 – 도구 활용 가이드 참조

5.9.1 왜 자동화된 지속적 통합이 권장되는가?

Tip 40. 지속적 통합이란 무엇인가?

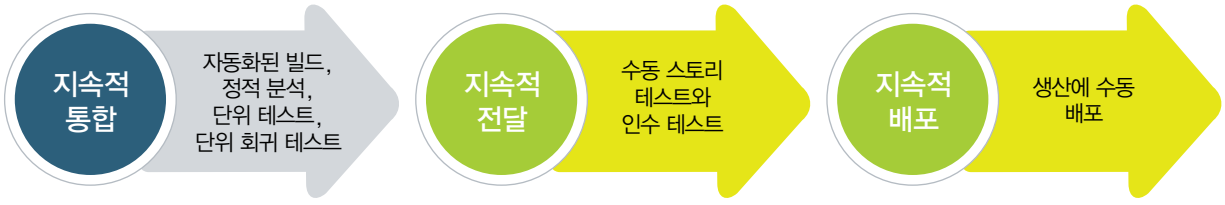
지속적인 통합: 새 코드가 공유된 저장소에 제공될 때마다 자동화된 빌드는 기존 코드와 통합될 수 있는지를 확인한다.

지속적인 통합 구축하기를 권장한다. 새 코드가 공유된 저장소에 더해질 때마다 자동화된 빌드는 단위 테스트와 회귀 테스트를 자동적으로 수행한다.

지속적 통합 프로세스는 다음과 같다.

1. 코드 체크 인(check in)
2. 자동화된 정적분석 수행
3. 빌드
4. 자동화된 단위 테스트와 회귀 테스트
5. 스토리 테스팅을 위한 테스트 서버에 배포

개발자들이 동시에 작업할 때, 지속적인 통합은 모든 개발자들이 변경과 수정이 반영된 최신 코드에 접속할 수 있다. 자동화된 빌드 후에는 스토리 테스팅을 빠르게 시작할 수 있다. 코드가 버전 관리에 커밋될 때마다 단위 테스트와 단위 회귀 테스트를 자동적으로 하루에 여러 번 빌드 할 수 있고, 수동 또는 자동화된 스토리 테스트와 인수 테스트 환경에 배포될 수 있다.



〈 도표 11. 지속적인 통합 〉

5.9.3 지속적 통합 준비

지속적 통합을 지원하기 위해서는 빌드와 단위 테스트 도구들을 구축할 필요가 있다. 구축할 때는 기존에 사용하고 있는 아래와 같은 도구 환경을 고려해야 한다.

- 빌드 도구
- 테스트 제어기
- 테스트 환경
- 단위 테스트를 테스트 환경에 연동
- CI 서버와 버전 관리 도구 설치
- 정적 분석 도구 설치

빌드 도구와 프로세스가 준비되면, 기존 도구와 환경에 빌드와 단위 테스트 워크플로를 구축할 필요가 있다.

- 빌드와 단위 테스트를 위한 스크립트와 워크플로 작성
- 단위 테스트 세트 작성
- 빌드와 단위 테스트가 잘 수행되고 있는지 확인
- 빌드 또는 테스트 상의 문제 해결

위 사항이 구축되면 지속적 통합을 수행할 수 있다.

5.9.4 빌드와 배포 개선하기

프로젝트에 권장되는 첫 단계는 자동화된 빌드와 테스트를 포함하는 지속적인 통합이다. 지속적 통합이 구축된 이후에, 빌드와 배포의 효과성과 효율성을 향상시키는 방법이 있는데 이것은 지속적 전달 및 배포에서 다루겠다.

5.10 시도해 볼 사항

5.10.1 지속적 전달 – 도구 활용 가이드 참조

지속적 전달 및 배포는 둘 다 매우 유용한 접근방법이다. 하지만 다른 프랙티스에 익숙해지고 난 뒤 수행하기를 권장한다.

특히, 지속적 전달을 수행하기 전에 지속적 통합이 반드시 성공적으로 수행돼야 한다. 지속적 전달은 지속적 베포를 수행하기 전에 반드시 성공적으로 수행돼야 한다.

지속적 배포를 도입하면 릴리즈 계획과 인수 테스팅을 변경해야 하며 다르게 수행해야 한다. 사용자는 인수 테스팅을 하지 않으며, 사용자의 요구에 대한 더 정확한 정보가 팀에 요구된다. 또, 더 정확한 요구사항과 인수 기준을 필요로 한다. 사용자는 팀을 신뢰할 필요가 있으며, 이러한 신뢰는 프로젝트 경험과정에서 생긴다.

Tip 41. 지속적 전달은 무엇인가?

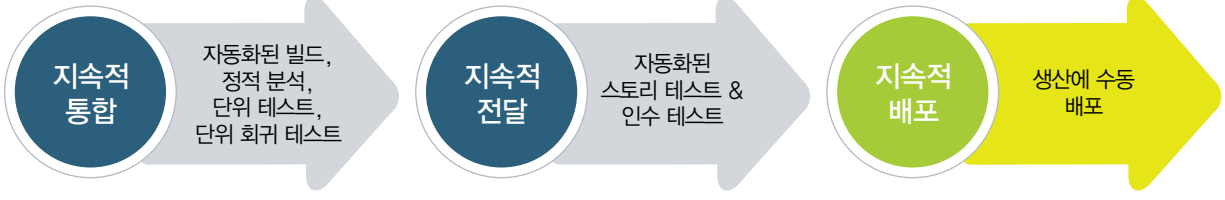
지속적 전달 : 코드가 저장소에 제출될 때마다 빌드되고, 해당 단위 테스트는 통과되며, 단위 회귀 테스트의 전체 세트 또한 통과된다. 회귀 테스트를 위해 빌드는 테스트 서버에 배포되며 회귀 테스트를 수동적으로 관리할 수 있다. 변경과 배포 리스크에 기반한 테스트에 대해 결정할 수 있다. 모든 것이 통과되면, 자동화된 배포가 시작된다. 배포를 위해 새로운 코드를 확인할 만큼 충분히 적합한 회귀 테스트 세트를 요구해야 한다. 최소 테스트 세트는 CI동안 수행되는 단위 테스트이며, 어떤 스토리와 인수 테스트를 자동화된 회귀 테스트에 추가할지 선택해야 한다.

지속적 전달은 지속적 통합의 다음 단계이다.

지속적 전달은 지속적인 통합에다가 릴리즈 프로세스까지 자동화되는 것이다. 생산으로의 배포는 수동이다.

지속적 전달이 구축되려면, 스토리 및 인수 테스트를 위한 자동화된 테스트 세트를 구축하고 유지할 필요가 있다. 지속적 전달을 지원하기 위해 인수 테스트 주도 개발을 사용할 필요가 있다.

지속적 전달의 목적은 배포가 빠르고 신뢰성 있고 반복 가능하며, 전달된 소프트웨어가 항상 릴리즈 가능한



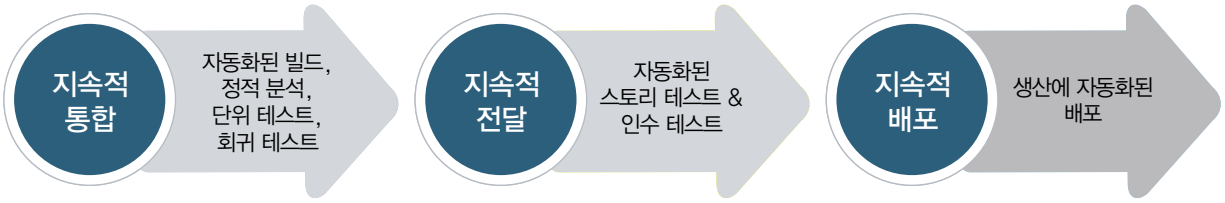
〈 도표 12. 지속적 전달 〉

5.10.2 지속적 배포 – 도구 활용 가이드 참조

Tip 42. 지속적 배포란 무엇인가?

지속적 배포 : 자동화된 체크는 문제를 식별하고 배포를 멈추고 수동 개입을 하는 데 사용된다. 빠른 타임 투 마켓(time to market)과 쉬운 롤백(rollback)이 가능한다. 지속적 전달의 성숙한 다음 단계가 지속적 배포이다.

지속적 배포는 지속적인 통합과 지속적 전달을 포함하며, 완전히 생산에 자동화된 배포를 포함한다.

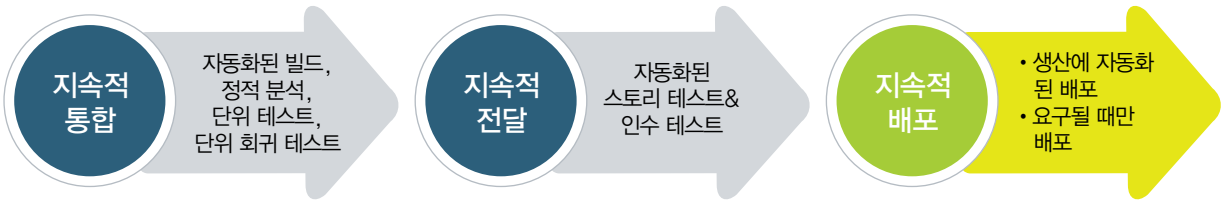


〈 도표 13. 지속적 전달 〉

지속적 통합과 지속적 전달이 구축되면 지속적 배포를 구출할 수 있다. 지속적 배포는 배포 활동을 더 빠르고 반복 가능하게 하며 원한다면 빈번하게 할 수 있다.

고객과 사용자가 지속적 배포가 없기를 바라는 이유가 있는데, 그 이유들은 규제적 제한, 날짜에 민감한 변경, 생산 제한, 변경 관리를 포함한다. 지속적 배포 구축은 IT 결정일 뿐 아니라 고객 및 사용자의 결정이다.

지속적 배포는 "요구에 따른 배포"라고 불린다. 왜냐하면 팀은 항상 배포할 준비가 돼 있지만 사용자가 배포를 요구할 때만 배포하기 때문이다.



〈 도표 14. 요구에 따른 배포 〉

5.10.3 기술 부채 및 리팩토링

기술 부채는 설계와 코딩 선택의 결과에 대한 개념이다. 만일 설계의 충분한 고려 없이 빠르게 수행하는 것을 선택한다면 구축한 설계와 코딩이 미래에는 잘 동작하지 않을 수도 있다. 프로젝트는 간결한 설계를 권장하지만, 이것이 설계를 대충해도 된다는 것을 의미하지는 않는다. 간결한 설계는 지금 필요한 것에 초점을 두지만, 미래를 위해서는 아무 생각도 하지 않아야 함을 의미하지는 않는다.

어떤 문제에 대한 단기적 수정으로 설계와 코딩 결정을 내린다면, 이러한 단기적 설계와 코딩을 대체하고 개선하는 계획을 세울 필요가 있다. 왜냐하면 지금 단기적 수정이 기술 부채가 될 수 있기 때문이다.
즉, 변경될 필요가 있는 것에 대해 작업할 시간이 없을 경우, 시간이 지남에 따라 해결하기가 더 어려워질 수 있다.
기술 부채가 있으면 리팩토링 이터레이션(재 설계 및 코드 리팩토링)을 할 필요가 있다. 새 코딩은 이전처럼 잘 동작하는지를 보장하기 위해 회귀 테스트를 할 필요가 있다. 리팩토링 활동은 다음을 포함한다.

- 코드 응집도 높이기
- 코드 결합도 낮추기
- 새 코드 테스트 (특히 단위 테스트, 인수 테스트는 그대로 유지)

설계와 코딩의 견고함을 보장하기 위해 리팩토링 이터레이션을 수행으로써 기술 부채를 피하는 것이 낫다.

Tip 43. 왜 리팩토링은 최고의 해결책이 아닌가?

기술 부채 상황에 놓이고, 리팩토링을 해야 하게 된다면, 사용자에게 직접적인 가치를 전달하지 못하는 이터레이션에서 시간을 보내게 된다. 항상 사용자에게 가치를 제공할 수 있는 것에 집중하라. 견고한 설계는 시간이 흐르면서 좋은 가치를 전달할 수 있다.

5.10.4 짝 프로그래밍

Tip 44. 왜 짝 프로그래밍인가?

짝 프로그래밍에서는 두 프로그래머가 하나의 컴퓨터를 공유한다. 이는 새로운 팀원을 참여시키거나 결함 해결 시 논의할 수 있어서 좋다. 짝 프로그래밍 방법으로 코드의 품질을 향상시킬 수 있다.

짝 프로그래밍은 유용하지만 선택적 활동이다. 전환될 수 있는 두 개의 역할이 있다.

- 드라이버는 코딩을 하는 역할로 무엇이 변경될 필요가 있는지를 전략적으로 살펴본다.
- 네비게이터는 리뷰하는 역할로 무엇이 달성될 필요가 있는지를 전반적으로 살펴본다.

짝 프로그래밍을 진행하는 동안 토론을 지속하는 것과 역할을 바꾸는 것이 중요하다.

짝 프로그래밍의 장점은 다음과 같다.

- 설계상의 어려움을 해결할 수 있다.
- 어려운 코딩을 해결할 수 있다.
- 유지보수를 위한 코드 리팩토링에 유용하다.
- 어려운 결함들을 디버깅할 수 있다.
- 지식과 기술을 공유할 수 있다.

5.10.5 짝 테스트

Tip 45. 왜 짝 테스트인가?

짝 테스트에서 두 사람은 테스트를 수행하기 위해 한 컴퓨터를 공유한다. 이것은 테스트 기술과 시스템 지식을 전달하므로 새로운 사람이 팀에 참여하기가 좋고 결함을 리뷰하는 데에도 좋다. 테스터와 개발자과 기술과 지식을 공유함으로 유용할 수 있다. 수정과 확인 테스트를 줄이고 싶을 때 더 효율적이다.

짝 테스트는 유용하지만 선택적 활동이다.
짝 테스트는 동일한 기계나 장비에서 두 사람이 함께 테스트를 설계 및 수행하는 접근법이다. 한 명은 테스트를 수행하는 것에 집중하며, 다른 한 명은 테스트를 위한 제안, 설계, 기록하는 것에 집중한다. 둘은 테스트를 수행하면서 아이디어, 테스트, 결과에 대해 토론한다.

짝 테스트는 다음과 같은 장점이 있다.

- 정보를 팀 내 새로운 사람에게 전달
- 테스트 코칭
- 탐색적 테스트 동안 아이디어를 공유하고 토론
- 결함 리뷰

짝으로 함께 일하는 것이 한 아이디어에 갇힐 가능성이 덜하기 때문에, 짝 테스트는 테스트의 창의성과 생산성을 향상시킬 수 있다.

5.11 참조 사항

5.11.1 기능 주도 개발 방법론(Feature-Driven Development)

▶ 기능 주도 개발 방법론

팀은 아키텍처 구조보다 기능을 구현한다. 기능들이 이미 충분히 통합되었으므로, 통합 단계에 대한 필요성이 제거된다.

기능 주도 개발(FDD)은 아래에 초점을 둔다.

- 사용자와 그들이 업무를 수행하는 데 필요로 하는 것
- 그들을 지원하는 데 필요한 구조
- 실용적이며 간결한 개발 프로세스

기능은 사용자에게 가치 있는 작은 단위이다. 하나 또는 그 이상의 사용자 스토리에 기술돼 있는 것을 발견하게 될 것이다. 기능들은 '액션-결과-오브젝트'(Action-Result-Object) 와 같은 형식에 기술될 수 있다.

예제 35. 액션-결과-오브젝트 형식의 기능

액션-결과-오브젝트 형식의 기능 예시

'마이 트래블'(MyTravel) 웹 사이트 - 기능: 총 비행 비용 계산

'오키드딜라이트'(OrchidDelight) 웹 사이트 - 기능: 선택한 부케 판매 허가

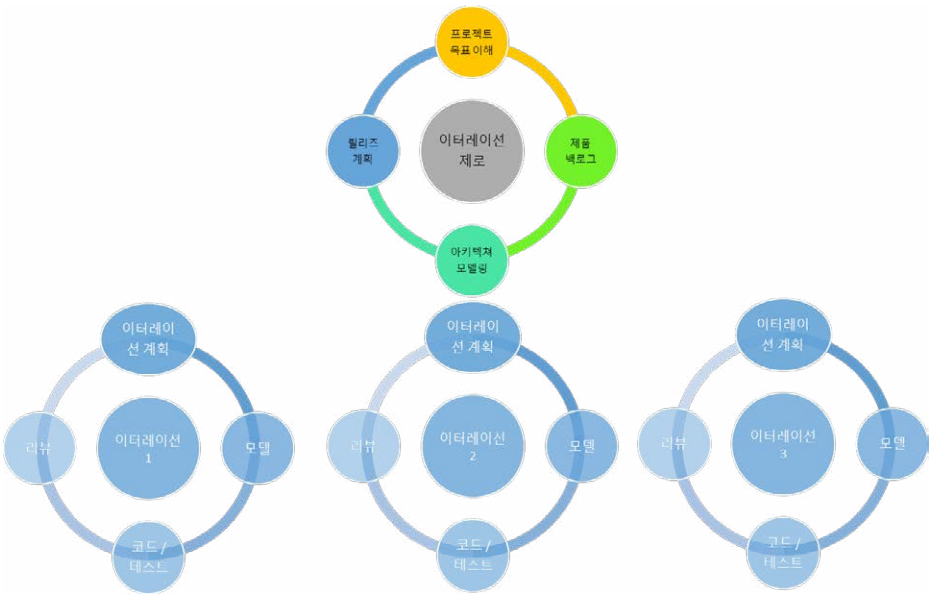
기능 주도 개발은 전반적인 릴리즈 계획과 제품 백로그가 계획되는 이터레이션 제로에서 시작한다. 일단 이터레이션이 시작되면 각 이터레이션에서 기능 주도 개발은 설계와 모델링에 기반해 개발한다.

이터레이션 제로에서 기능 주도 개발은 다음을 수행한다.

- 모델링 되고 있는 도메인에 대한 이해
- 제품 백로그와 릴리즈 계획 추가할 기능 목록 작성
- 소프트웨어 시스템의 아키텍처 모델 개발

이터레이션에서 기능 주도 개발은 다음을 수행한다.

- 이터레이션 계획 동안 어떤 기능이 구현될 지 결정하고 빠르게 모델링
- 설계와 개발 동안 기능, 코딩 및 단위 테스트에 대한 모델링
- 코딩, 단위 테스트
- 모델링 결과물과 코딩에 대한 리뷰



〈 도표15. 스크럼 개발 프로세스 〉

5.11.2 테스트 주도 개발(Test-Driven Development)

테스트 주도 개발은 코딩 전에 단위 테스트 세트를 먼저 작성하고 자동화하는 접근법이다. 단위 테스트 세트를 통과하는 코드를 나중에 개발한다. 테스트 주도 개발 프로세스는 아래와 같다.

1. 변경 또는 새로운 기능에 대한 테스트 코드를 자동화된 테스트 코드에 추가한다.
2. 자동화된 테스트 코드를 수행한다.
3. 기존 테스트 코드가 통과하는지 확인한다.
4. 새로운 테스트가 실패하는지 확인한다. 코딩이 아직 완성되지 않았기 때문인데 만약 통과됐다면 충분한 테스트 코드가 아니거나 이 요구사항에 대해서는 코딩이 완료된 것이다.
5. 새로운 테스트 코드를 통과시키는 코딩을 작성한다.
6. 모든 테스트를 수행하고 통과하는지 확인한다.
7. 결함을 수정한다.
8. 모든 테스트를 수행하고 통과하는지 확인한다.
9. 변경 또는 새로운 기능에 대한 코드를 완성하고 모든 테스트 코드가 통과할 때까지 반복한다.

테스트 주도 개발의 장점은 아래와 같다.

- 자동화된 단위 테스트는 회귀 테스트가 용이하다.
- 코드 커버리지가 달성된다.
- 불필요한 코드가 덜 작성된다.
- 디버깅이 일반적으로 쉬워진다.
- 테스트 주도 개발은 결함들을 방지할 수 있다.

테스트 주도 개발의 단점은 아래와 같다.

- 테스트 요구사항에 대해서만 테스트 코드가 작성된다.
- 네거티브 테스트가 덜 강조된다.
- 개발 코드와 자동화된 테스트 코드의 리팩토링이 증가된다.
- 테스트 코드 작성으로 비용이 들 수 있다.
- 어떤 기술은 잘 지원되지 않는다.
- 프로그래머들이 이 접근법을 사용하도록 설득하는 것이 어려울 수 있다.
- 개발 코드는 테스트를 통과한 만큼만 좋다.

5.11.3 행동 주도 개발(Behaviour-Driven Development)

행동 주도 개발은 코딩 전에 작성하는 것이 단위 테스트 코드가 아니라 스토리와 인수 테스트 세트 라는 점을 제외하고는 테스트 주도 개발과 유사하다. 가능하다면 테스트는 반드시 자동화돼야 하며, 특히 행동 주도 개발을 지원하는 설계 도구가 이용 가능해야 한다.

행동 주도 개발의 프로세스는 아래와 같다.

1. 사용자 스토리로부터 BDD 도구의 입력으로 적절한 BDD 테스트 세트를 작성한다.
2. 자동화된 인수 테스트에 테스트 세트를 추가한다.
3. 자동화된 인수 테스트를 수행한다.
4. 기존 테스트 코드가 통과하는지 확인한다.
5. 새로운 테스트가 실패하는지 확인한다. 코딩이 아직 완성되지 않았기 때문인데 만약 통과됐다면 충분한 테스트 코드가 아니거나 이 요구사항에 대해서는 코딩이 완료된 것이다.
6. 새로운 테스트 코드를 통과시키는 코딩을 작성한다.
7. 모든 테스트를 수행하고 통과하는지 확인한다.
8. 결함을 수정한다.
9. 모든 테스트를 수행하고 통과하는지 확인한다.
10. 변경 또는 새로운 기능에 대한 코드를 완성하고 모든 테스트 코드가 통과할 때까지 반복한다.

5.11.4 인수 테스트 주도 개발(Acceptance Test-Driven Development)

인수 테스트 주도 개발은(ATTD) 인수 테스트에 사용될 코드를 작성한다는 점을 제외하면 행동 주도 개발(BDD) 접근법과 유사하다.

CHAPTER 06

작업 산출물 작성 방법

6.1 프로젝트 계획서

프로젝트 계획서

■ 목적

프로젝트 목표 및 제약 조건에 대한 1 페이지 분량의 설명

■ 설명

고객, 제품 책임자 및 팀 리더가 합의한 프로젝트의 주요 목표에 대한 간략한 설명이다.
이것은 항상 볼 수 있는 곳에 있어야 하고 종종 그래픽 형식으로 돼 있으며 팀의 사무실 벽에 게시된다. 도표, 마인드맵, 표 및 목록 등 모든 적절한 형식이 사용될 수 있다. 사람들이 프로젝트의 상위 방향성에 대해 확인하고 상기할 수 있도록 해 주는 유용한 도구이다.

■ 활동

- 전제 조건 및 이터레이션 제로

■ 역할

- 고객
- 제품 책임자
- 팀 리더

프로젝트 계획서 목적	1 페이지 분량의 프로젝트 목표: 목표, 리스크, 제약 조건, 대상 및 성공 지표. 프로젝트 계획서는 프로젝트의 이유가 한 페이지에 요약돼 있고 다음에 대한 간결한 설명을 포함한다. • 식별된 목표 • 프로젝트를 완료하지 못하는 식별된 리스크 • 프로젝트로부터 기대되는 이익 • 식별된 제약 조건 (예: 예산 및 납기) • 식별된 성공 지표
역할과 책임	고객이 프로젝트 계획서 초안을 작성한다. 팀 리더와 제품 책임자가 이에 동의하면 이것은 방향에 대한 가이드가 된다. 프로젝트의 방향이 크게 바뀌면 제품 책임자와 팀 리더가 헌장 변경에 합의하고 팀 리더는 변경 사항을 작성해 팀이 새 프로젝트 계획서를 파악하게 한다. 제품 책임자는 먼저 고객이 변경 사항을 알고 있으며 동의하는지 확인해야 한다.
보관 / 게시	포스터, 스크린 세이버, 배너 – 팀 사무실의 벽면에 볼 수 있도록 게시 한다.
활동	프로젝트 계획서 초안은 고객에 의해 프로젝트 시작의 전제 조건으로서 작성되며 이터레이션 제로 동안 합의된다.

〈 **표 3. 프로젝트 계획서 요약** 〉

프로젝트 계획서는 간결해야 하며 모든 팀원이 볼 수 있어야 하고 명확해야 한다. 마인드맵, 포스터 등은 프로젝트 계획서 문서의 좋은 예이다.

예제 36. 플립차트에 게시된 목록 상의 '마이트래블(MyTravel)' 프로젝트 목표들

화이트보드에 쓰여진 프로젝트 계획서 고객 광조(CEO)

CEO '광조'는 웹사이트가 여름 휴가철의 최다 구매 성수기를 대비해 업데이트되기를 바란다. 그는 마케팅 일정에 맞는 출시를 원한다.

프로젝트 목표

• '마이트래블(MyTravel)' 을 가족의 여름철을 위한 휴가 선택 주요 웹사이트로 만들기
• 휴가철을 위해 온라인 검색과 구매가 가능하도록 하기
• 고객의 서비스 비용 줄이기

프로젝트 목표를 달성하지 못할 경우의 리스크

• 500,000,000원으로 추정되는 매출 손실
• 경쟁사가 온라인 서비스로 제공하는 250,000,000원으로 추정되는 시장 점유율 손실

기회

• 300,000,000원만큼의 매출 증가
• 1년에 200,000,000원만큼의 비용 감소
• 12% 수익 상승

제약

- 제품 책임자인 '숙자'는 '마이샵'(MyShop)과도 작업할 예정이어서 일부 시간만 가능함
- 작업 예산은300,000,000원

목표

- 12월 1일 전에 웹사이트 완료
- 마케팅 일정에 맞춰 마케팅이 요청하면 웹사이트가 론칭해야 함

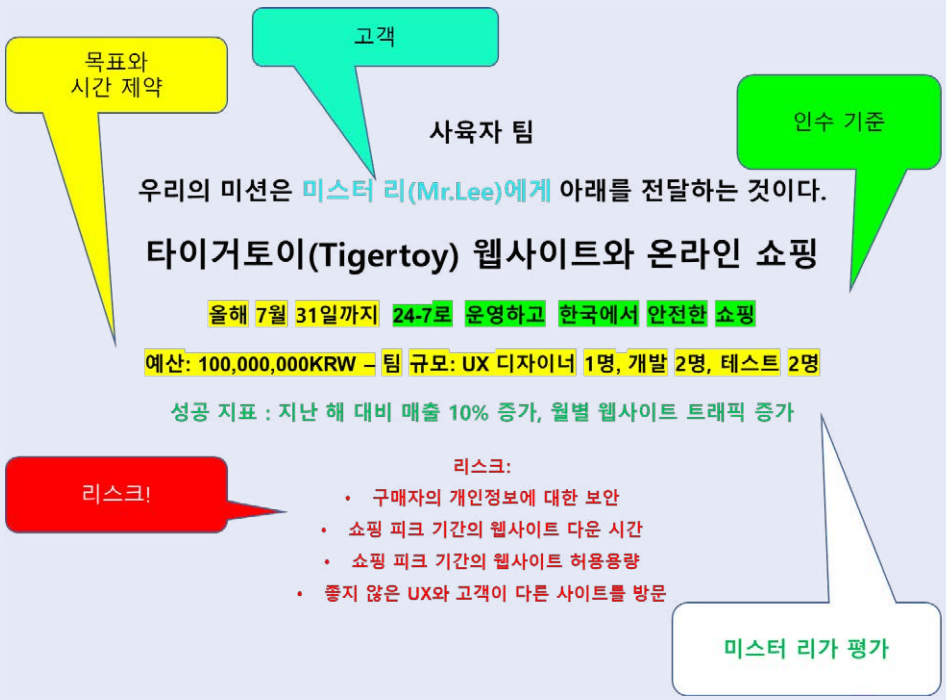
성공 지표

- 매출/수익 증가
- 비용 감소
- 시장 점유율 유지 또는 증가, 고객으로부터의 좋은 피드백

포스터로서의 프로젝트 계획서

'타이거 토이즈'(TigerToys) 웹사이트와 온라인 쇼핑 프로젝트 팀은 팀 룸 벽에 부착할 포스터에 프로젝트 계획서를 만든다.

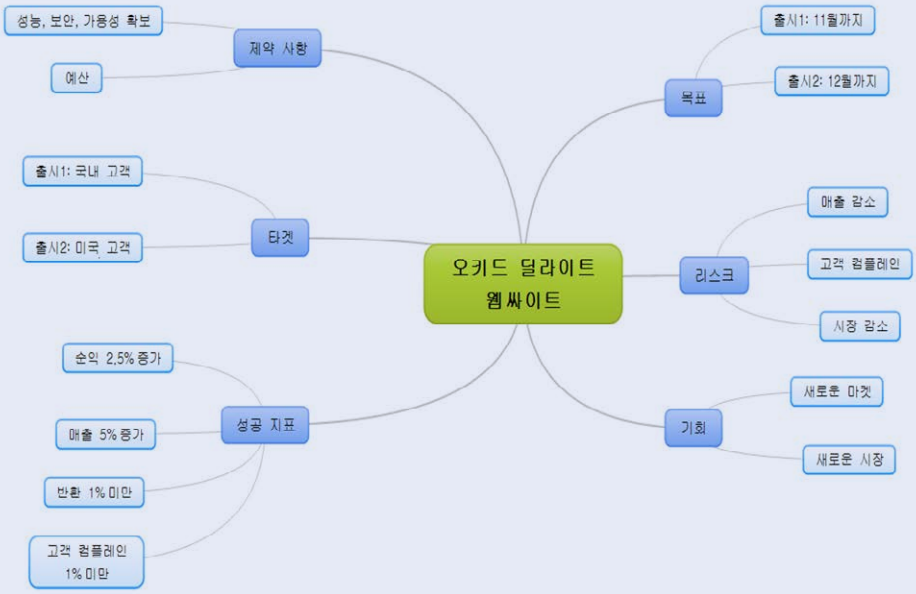
예제 37. '타이거 토이즈'(TigerToys) 프로젝트 계획서(포스터)



프로젝트 계획서 마인드맵

'오키드딜라이트' 웹 쇼핑 팀은 고객 '아름'을 만났을 때 마인드맵을 만들었다. '아름'의 목표를 기억하기 위해 마인드맵을 사용한다.

예제 38. '오키드딜라이트' 프로젝트 계획서(마인드맵)



6.2 제품 백로그

제품 백로그

■ 목적

구현을 위한 작업 항목 목록

■ 설명

제품 백로그에는 현재의 이터레이션 이후 팀이 수행할 작업을 설명하는 사용자 스토리들이 포함돼 있다. 이터레이션 계획이 수행될 때 이터레이션을 위한 완전한 사용자 스토리들이 제품 백로그에 충분히 있어야 한다.
제품백로그는 제품 책임자가 담당한다.

■ 활동

- 제품 백로그 관리
- 이터레이션 계획

■ 역할

- 제품 책임자
- 팀

제품 백로그 목적	제품 백로그에는 수행될 이터레이션에 아직 할당되지 않은 모든 작업 항목이 있다. 사용자의 요구 사항과 필요 사항 팀으로부터의 기술적 요구 사항과 필요 사항 해결해야 할 결함
역할과 책임	제품 책임자가 제품 백로그를 관리한다. 제품 책임자는 제품 백로그에 작업 항목을 추가하고, 제거하고, 우선 순위를 변경할 수 있다.
보관 / 게시	제품 백로그는 사용자 스토리 카드 세트로 유지돼야 하며 제품 백로그가 제품 책임자에 의해 업데이트되지 않을 때에는 팀의 사무실에 있는 박스에 보관된다.
활동	제품 백로그는 이터레이션 제로에서 처음 작성되며 이터레이션 계획 미팅들 사이에 제품 책임자에 의해 계속 업데이트된다. 다음 이터레이션에서 수행될 수 있도록 제품 백로그를 확인해야 한다. 제품 백로그의 크기를 최소한으로 유지한다.

〈 표 4. 제품 백로그 요약 〉

제품백로그는 사용자 스토리 형태로 작성되며 카드 형태로 보관되거나 Excel 이나 구글닥스 같은 도구에 기록될 수 있다. (6.6사용자 스토리 참조)

6.3 릴리즈 계획서

릴리즈 계획서

■ 목적

어떤 기능이 언제 인도될 지를 설명

■ 설명

이터레이션 제로 동안 릴리즈 계획서가 작성되고, 요구 상황이 변경되었거나 릴리즈 계획이 연장돼야 하는 경우 업데이트된다. 상위 수준에서는 각 릴리즈에서 기대되는 일반적인 기능을 설명하고 보다 상세한 수준에서는 사용자 스토리를 각 릴리즈에 할당한다.
릴리즈 계획서는 제품 책임자가 담당하며 고객 및 팀 리더와 협력한다.

■ 활동

- 이터레이션 제로
- 릴리즈 계획

■ 역할

- 고객
- 제품 책임자
- 팀 리더

릴리즈 계획서의 목적	릴리즈 계획서의 목적은 릴리즈에 필요한 기능을 설명하는 것이다. 고객이 사용자에게 릴리즈 되는 일정을 알고 싶을 때 릴리즈 계획서가 필요하다.
역할과 책임	제품 책임자는 팀 리더의 도움을 받아 고객의 요구 사항을 반영해 릴리즈 계획서를 작성한다.
보관 / 게시	릴리즈 계획서가 작성되고 보관되는 방법은 프로젝트에 따라 다르다. 예를 들어, 도구에 보관하거나 스프레드 시트 상에 보관하거나 화이트보드 상의 목록에 보관할 수 있다.
활동	릴리즈 계획서는 릴리즈 계획(Release Planning) 중에 작성된다.

〈 **표 5. 릴리즈 계획서 요약** 〉

팀의 이터레이션 평균 속도를 고려하여 전체 릴리즈 일정을 추정하고 나서 각 이터레이션별로 스토리들을 할당한다. 이터레이션 초기에는 팀의 평균 속도보다 적은 양을 할당하고 이후 늘려간다. 마지막 이터레이션은 보통 통합테스트와 버퍼로 확보한다. 버퍼 스프린트는 각 이터레이션별로 완료하지 못한 스토리들을 처리하기 위한 것이다.

이터레이션 1	이터레이션 2	이터레이션 3	이터레이션 4	이터레이션 5	이터레이션 6	이터레이션7
S □ , S □ S □,	S □ , S □ S □, S □	S □ , S □ S □ , S □ S □	S □ , S □ S □ , S □ S □ , S □	S □ , S □ S □ , S □ S □ , S □	통합테스트	버퍼

〈 **도표16. 릴리즈 계획 상황판** 〉

6.4 릴리즈 번-업 차트

릴리즈 번-업 차트

■ 역할

계획된 릴리즈 진행 상황을 추적하는 수단 제공

■ 설명

릴리즈 번-업 차트는 완료된 작업 및 계획된 전체 작업과 관련해 계획된 릴리즈를 위한 프로젝트 진행 상황을 시각적으로 제공한다. 프로젝트가 릴리즈를 위해 진행될 때 완료된 작업은 증가하며 계획된 전체 작업은 이상적으로는 동일하게 유지돼야 하지만 변경되는 경우 차트 상에 명확하게 시각화 된다.
완료된 작업 및 계획된 전체 작업에는 사용자 스토리의 크기를 추정하는 데 사용되는 것과 동일한 단위가 사용돼야 한다.
릴리즈 번-업 차트는 팀 리더가 담당한다.

■ 활동

- 릴리즈 계획 (Release Planning)

■ 작업 산출물

- 팀 리더
- 팀원
- 제품 책임자
- 고객

목적	다음 릴리즈를 위한 진행 상황을 모든 사람이 볼 수 있게 하는 것
역할과 책임	팀 리더가 릴리즈 번-업 차트를 담당한다. 모든 사람이 이것을 통해 다음 릴리즈를 위한 진행 상황을 알 수 있다.
보관 / 게시	도구를 사용해 작성할 수 있지만 팀 사무실의 벽면에 볼 수 있도록 게시돼야 한다.
활동	릴리즈 번-업 차트는 릴리즈 계획 후에 만들어지고 제품 책임자가 인도 가능한 코드를 인수하는 인수 테스팅 기간 후에 업데이트된다.

〈 표 6. 릴리즈 번-업 차트 요약 〉

6.4.1 책임 및 시간 설정

릴리즈 번-업 차트는 진행상황을 보여주기 위해 팀 리더에 의해 릴리즈 계획에 기반해서 만들어진다. 동작하는 소프트웨어가 사용자에게 전달될 때나 제품 책임자가 사용자 스토리들을 추가할 때 업데이트 된다.

릴리즈 번-업 차트는 다음을 포함한다.

- 다음 릴리즈에 원래 계획된 작업의 양
- 다음 릴리즈에 계획된 이상적인 진행상황
- 지금까지의 실제적 진행상황
- 프로젝트가 뒤쳐졌는지, 앞서가고 있는지
- 프로젝트에 작업이 추가되었거나 제거되었는지

6.4.2 차트 게시

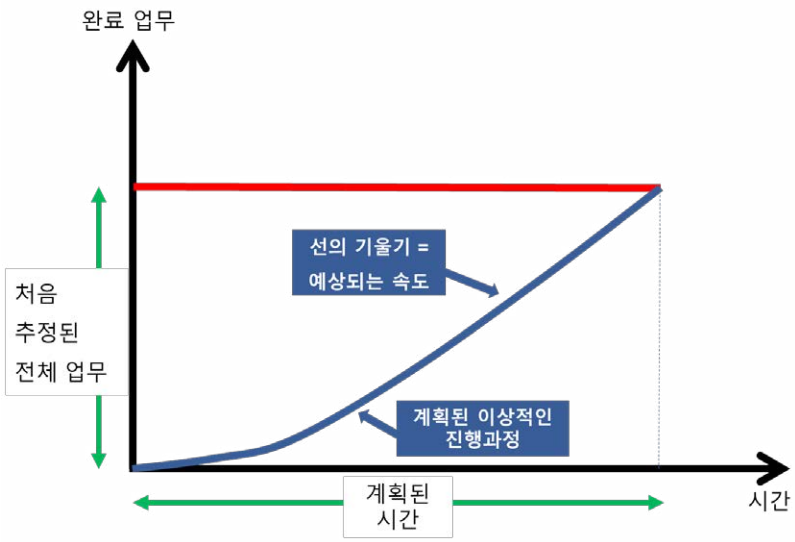
팀이 함께 있으므로, 팀 사무실 벽에 놓인 차트는 효과가 좋다. 차트는 다음과 같이 게시한다.

- 화이트보드에 게시
- 벽에 차트가 표시된 스프레드시트나 다른 도구로 게시

제품 책임자, 사용자, 고객이 팀을 방문하면 벽에 있는 차트를 볼 수 있다.

6.4.3 차트 활용

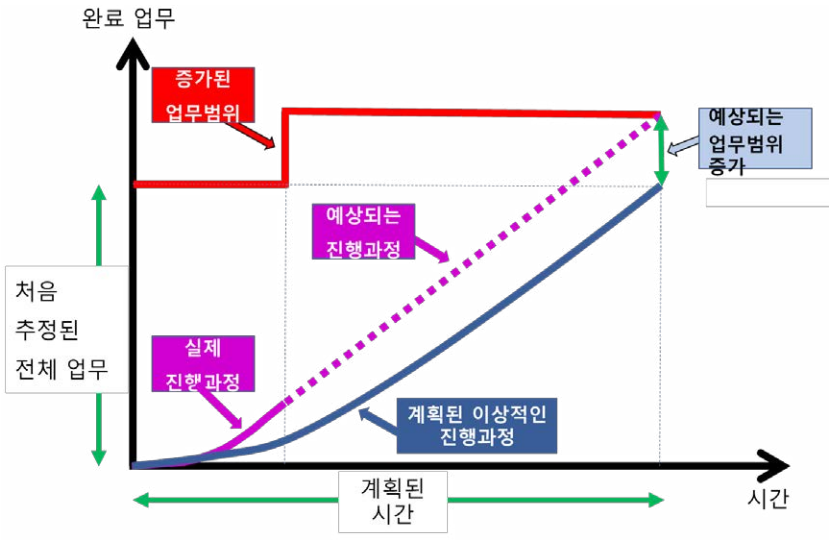
초기의 번-업 차트는 릴리즈 계획으로부터 예상된 팀 속도에 기반해 작성될 수 있다.



〈 차트 1. 번-업 차트 - 초기 〉

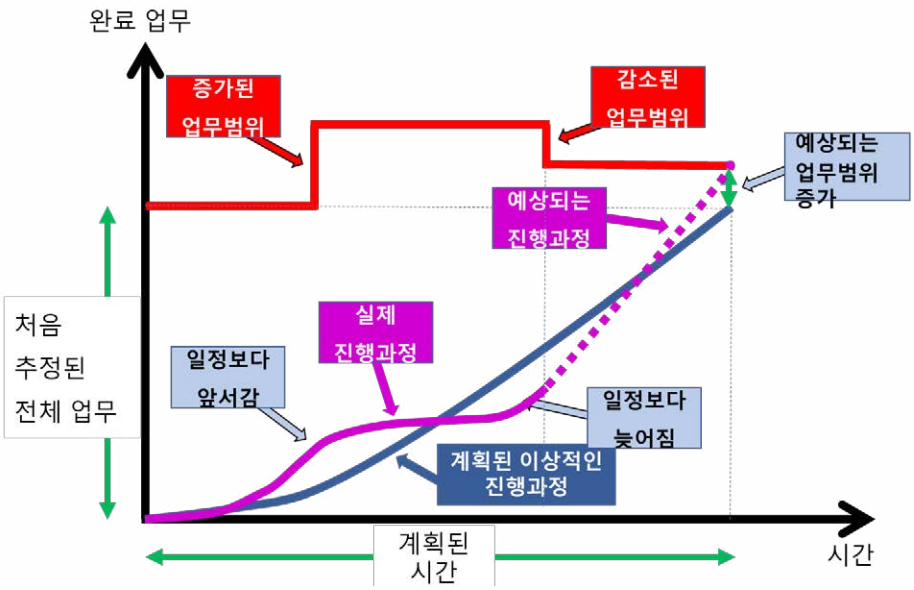
가로축은 이터레이션의 계획된 총 기간을 보여주며, 세로축의 붉은 선은 추정된 전체 작업량을 보여주며 스토리 포인트를 사용한다. 파란 줄은 릴리즈 처음부터 끝까지의 이상적인 진행상황을 보여준다. 이 파란선의 기울기는 팀의 예상 속도를 나타낸다. 처음에는 팀 학습을 반영해 천천히 출발한다. 이 차트는 첫 릴리즈에서 나중 릴리즈까지 예상 속도가 일정하므로 이상적인 진행이다.

이터레이션이 진행됨에 따라, 팀 리더는 실제 진행상황에 대한 정보를 차트에 추가한다. (보라색 선)



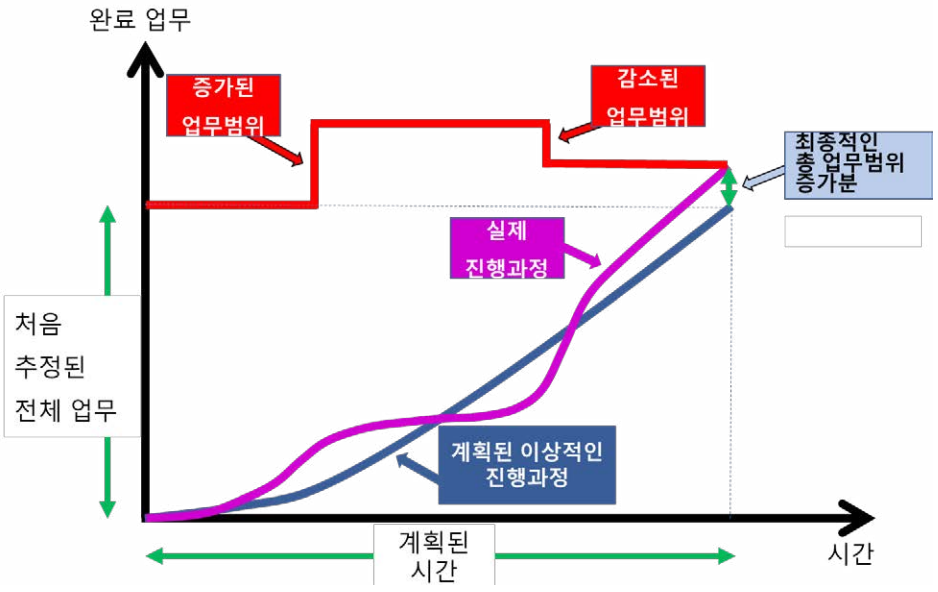
〈 차트 2. 릴리즈 진행상황과 업무 범위 증가를 보여주는 릴리즈 번-업 차트 〉

위 차트에서 실제 진행상황은 예상을 초과하며 릴리즈 시점까지 처음 계획된 것 보다 더 많은 작업이 팀에 의해 수행될 것이라고 예상할 수 있다. 고객과 제품 책임자는 팀 리더의 동의를 받아 릴리즈 작업량을 증가시키기로 결정한다. 번-업 차트의 장점은 작업 기울기 변경을 명쾌하게 보여준다는 것이다.
몇 개의 이터레이션 후에, 예상된 진행상황을 벗어났으며, 심지어 릴리즈 시작 시점에 예상됐던 원래 계획보다 뒤쳐져 있다.



〈 차트 3. 업무범위 감소와 변동하는 진행상황을 보여주는 릴리즈 번-업 차트 〉

위에서 보이는 대로, 이후의 상황 예측은 비록 초반의 기대를 충족하지 못하더라도 여전히 원래 릴리즈에 계획된 것 보다 더 많은 기능을 완료할 수 있음을 시사한다. 그러므로 증가한 범위는 현재 예상된 진행상황을 맞추기 위해 감소하게 된다.
릴리즈 시점까지, 팀이 원래 계획된 업무 범위를 약간 초과했고 문제를 성공적으로 극복한 것을 확인할 수 있다.



〈 차트 4. 릴리즈와 완료된 업무량을 통해 진행상황을 보여주는 릴리즈 번-업 차트 〉

6.4.4 차트 작성

릴리즈 번-업 차트를 작성하기 위해서는 아래와 같은 데이터 표가 필요하다.

- 릴리즈 날짜의 진행상황을 보여주는 릴리즈 계획에 기반한 데이터 표

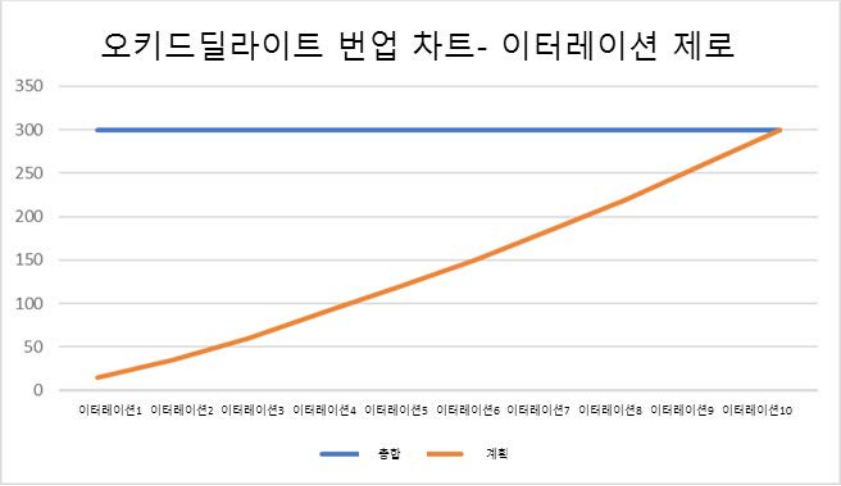
데이터 표 예시: 릴리즈 번-업 차트를 작성하는 이터레이션 제로

300스토리 포인트, 10개의 이터레이션을 릴리즈할 계획을 가정하면 첫 데이터 표는 다음과 같다.

이터레이션	이터레이션 1	이터레이션 2	이터레이션 3	이터레이션 4	이터레이션 5	이터레이션 6	이터레이션 7	이터레이션 8	이터레이션 9	이터레이션 10
총합	300	300	300	300	300	300	300	300	300	300
계획	15	20	25	30	30	30	35	35	40	40

〈 표 7. 프로젝트 초기 번-업 차트 데이터 표 〉

처음에는 천천히 시작하지만 이터레이션 9, 10에서는 40 스토리 포인트까지 증가했다.
초기 번-업 차트는 아래와 같다.



〈 차트 5. 오키드딜라이트 초기 번-업 차트 〉

6.4.5 차트 업데이트하기

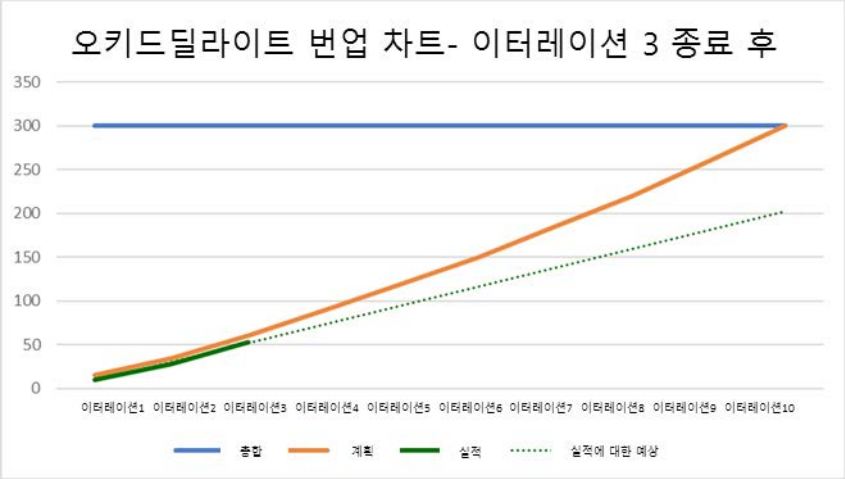
세 개의 이터레이션 후의 차트

세 개의 이터레이션을 진행한 후 데이터 표는 아래와 같다.

이터레이션	이터레이션 1	이터레이션 2	이터레이션 3	이터레이션 4	이터레이션 5	이터레이션 6	이터레이션 7	이터레이션 8	이터레이션 9	이터레이션 10
총합	300	300	300	300	300	300	300	300	300	300
계획상	15	20	25	30	30	30	35	35	40	40
완료	10	18	25							

〈 표 8. 세 개의 이터레이션 후 번-업 차트 데이터 표 〉

이 데이터는 번-업 차트에 업데이트 된다.



〈 차트 5. 오키드딜라이트 초기 번-업 차트 〉

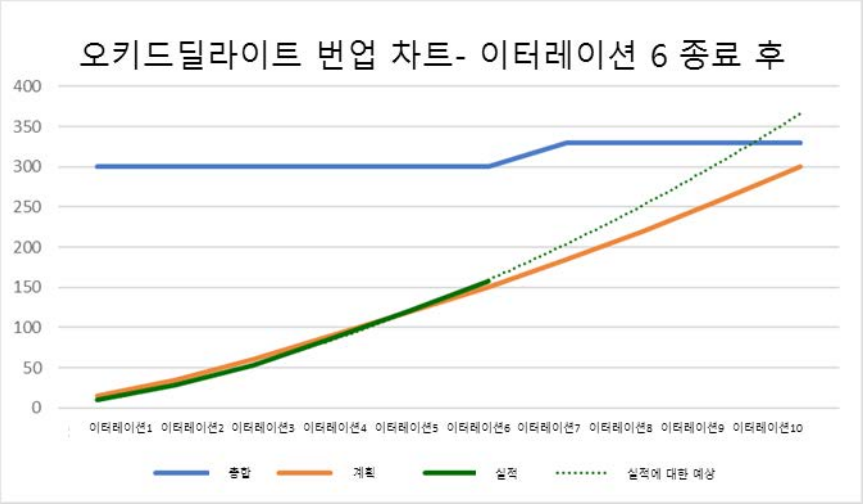
일정이 뒤쳐졌음을 확인할 수 있다. 선의 추세는 300 스토리 포인트 목표를 달성 못할 것으로 예상된다. 그러나 일정이 뒤쳐진 이유가 팀 내 일시적 병가 이었기 때문에 앞으로는 속도가 증가할 것을 기대한다.
계획상과 실제간 차이의 원인을 캡션으로 차트에 추가하는 것은 좋은 활동이다. 표로 하거나 차트 제목에 주석을 달 수 있다.

이터레이션 6 이후의 차트

이터레이션 6의 끝에, 차트로부터 프로젝트가 일정보다 앞섰다는 것을 확인할 수 있고 원래 목표를 초과할 수 있음을 추세선으로 예측할 수 있다. 이로 인해 추가적 작업이 계획에 더해졌다. 파란색 '총합' 선이 상승했다.

이터레이션	이터레이션 1	이터레이션 2	이터레이션 3	이터레이션 4	이터레이션 5	이터레이션 6	이터레이션 7	이터레이션 8	이터레이션 9	이터레이션 10
총합	300	300	300	300	300	300	330	330	330	330
계획상	15	20	25	30	30	30	35	35	40	40
완료	10	18	25	33	35	37				

〈 표 9. 이터레이션 6 이후의 번-업 차트를 위한 데이터 표 〉



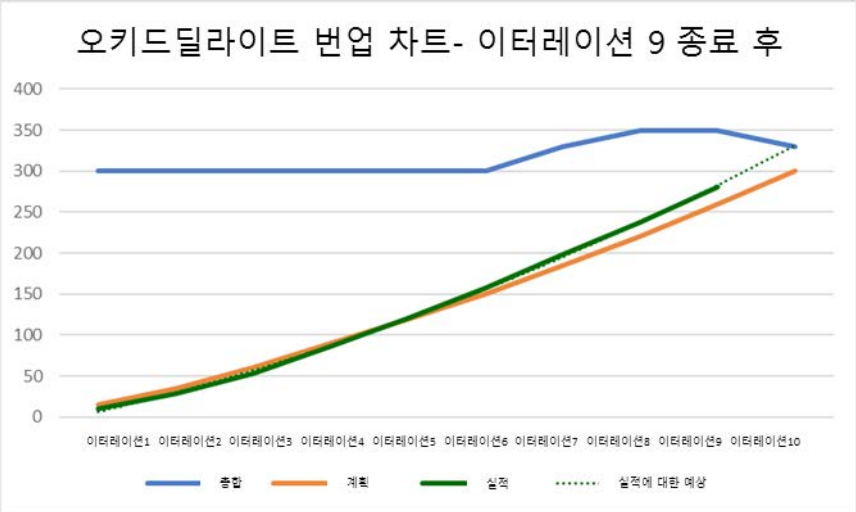
〈 차트 7. 이터레이션 6 이후의 오키드딜라이트 번-업 차트 - 작업이 추가됨 〉

이터레이션 9 이후의 차트

프로젝트의 일정이 계획보다 앞서므로, 이터레이션 7 끝에서 계획상 작업량보다 더 많은 작업이 추가되었다는 것을 알 수 있다. 원래 일정보다는 앞섰지만, 속도가 지속적으로 증가하지는 않았으며, 이터레이션8과 9에서 완료된 실제 작업이 새로운 목표를 달성하리라고 예상하는 것 보다는 덜하였다.

이터레이션	이터레이션 1	이터레이션 2	이터레이션 3	이터레이션 4	이터레이션 5	이터레이션 6	이터레이션 7	이터레이션 8	이터레이션 9	이터레이션 10
총합	300	300	300	300	300	300	330	350	350	325
계획상	15	20	25	30	30	30	35	35	40	40
완료	10	18	25	33	35	37	40	40	43	

〈 표10. 이터레이션9 이후의 번-업 차트를 위한 데이터 표 〉



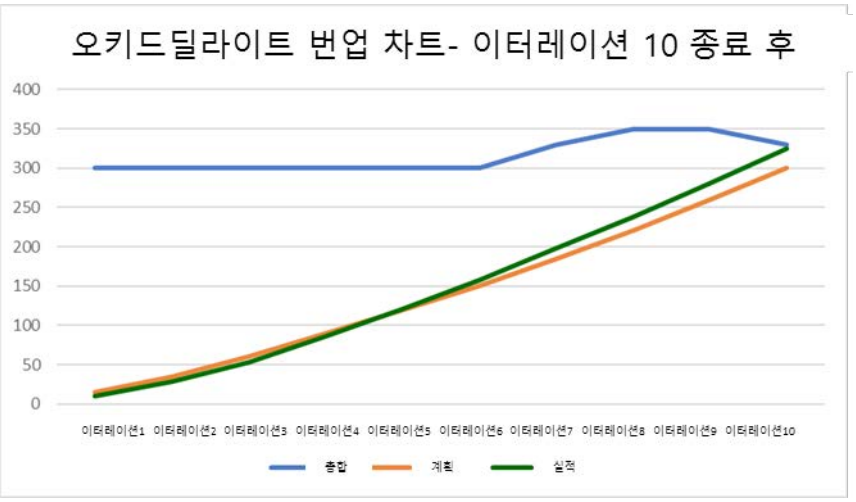
〈 차트 8. 이터레이션9 이후의 오키드딜라이트 번-업 차트 〉

이터레이션10 이후의 차트 – 마지막 이터레이션

마지막 이터레이션에서는, 팀 리더, 제품 책임자, 사용자, 고객이 지금까지 일어났던 일들에 관해 논의하며 작업량을 약간 줄이는 것에 동의한다. 프로젝트는 여전히 원래 계획된 것 이상을 완료할 것이다.
차트에서 보이는 대로, 릴리즈는 원래 계획보다 25이상인 325 스토리 포인트의 가치에 달하는 스토리를 완료했다.

이터레이션	이터레이션 1	이터레이션 2	이터레이션 3	이터레이션 4	이터레이션 5	이터레이션 6	이터레이션 7	이터레이션 8	이터레이션 9	이터레이션 10
총합	300	300	300	300	330	330	350	350	350	325
계획상	15	20	25	30	30	30	35	35	40	40
완료	10	18	25	33	35	37	40	40	43	44

〈 표11. 이터레이션10 이후의 프로젝트를 위한 데이터 표 〉



〈 차트 9. 이터레이션10 이후의 오키드딜라이트 번–업 차트 – 작업이 감소됨 〉

6.5 이터레이션 백로그

이터레이션 백로그

■ 목적

현재의 이터레이션 동안 구현될 작업 항목 목록

■ 설명

이터레이션 백로그에는 현재의 이터레이션 동안 팀이 수행할 작업을 설명하는 사용자 스토리가 포함돼 있다. 사용자 스토리는 제품 책임자가 선택한다. (그리고 팀 리더와 합의한다).
추가적인 사용자 스토리는 기술 작업을 지원하고 사용자 스토리들 사이의 의존성에 맞추기 위해 팀이 제안할 수 있다.
해당 이터레이션이 시작될 때 이터레이션 백로그를 구현하는 데 필요한 공수는 이터레이션 동안 팀에 기대되는 공수에 가까워야 한다. 결코 초과해서는 안 된다.

■ 활동

- 이터레이션 계획
- 개발 및 테스팅

■ 작업 산출물

- 제품 책임자
- 팀 (팀 리더 포함)

목적	해당 이터레이션에 수행될 작업에 동의하는지 확인하기
역할과 책임	제품 책임자와 팀 리더는 팀의 조언을 받아 이터레이션 백로그에 합의한다.
보관 / 게시	이터레이션 백로그는 사용자 스토리 카드 세트여야 하며 이터레이션 시작 시에 구현될 스토리를 위해 준비된 작업상황판의 해당 부분에 배치된다.
활동	이터레이션 백로그는 이터레이션 계획 동안 결정된다.

〈 표 12. 이터레이션 백로그 요약 〉

6.6 사용자 스토리

사용자 스토리

■ 목적

시스템의 필수 기능 설명

■ 설명

사용자 스토리는 인도될 시스템의 기능에 대한 설명을 제공한다. 요구 사항을 완전히 파악하는데 필요한 추가적인 토론이 그 목적이다. 대부분의 사용자 스토리는 사용자 요구 사항을 설명한다. 그러나 어떤 스토리는 다른 사용자 스토리들을 지원하는 데 필요한 작업(예: 사용자는 모르지만 시스템에 필요한 기술적 측면의 구현) 또는 인도될 코드의 수정을 설명한다.
사용자 스토리는 다양한 이해 관계자가 생성 할 수 있으며 제품 책임자의 책임이다.

■ 활동

- 이터레이션 제로
- 제품 백로그 관리
- 개발 및 테스팅

■ 작업 산출물

- 제품 책임자
- 사용자
- 팀

목적	사용자가 요구하는 것으로, 일반적으로 소프트웨어의 기능 설명
역할과 책임	제품 책임자는 일반적으로 사용자 요구 사항을 반영하는 사용자 스토리를 작성하고 팀원은 기술적 스토리를 작성한다. 결함 해결을 위한 사용자 스토리는 다양한 이해 관계자가 작성할 수 있다.
보관/ 게시	사용자 스토리는 A5 크기의 카드에 기록돼야 한다. 각 사용자 스토리는 "역할-기능-이유"의 형식을 가지며, 우선 순위, 크기 및 (알려진 경우) 릴리즈를 위한 공간이 있다. 카드의 뒷면에는 인수 기준이 있다.
활동	사용자 스토리는 제품 백로그 관리 중에 제품 책임자가 작성하지만 다른 이해 관계자도 언제든지 사용자 스토리를 작성할 수 있다.

〈 표 13. 사용자 스토리 요약 〉

6.6.1 사용자 스토리 포맷

사용자 스토리는 새로운 사용자 요구사항 또는 변경요구를 최소한의 기록하는 빠른 방법이다.
요구되는 기능은 “스토리 카드” 형태로 기록된다.

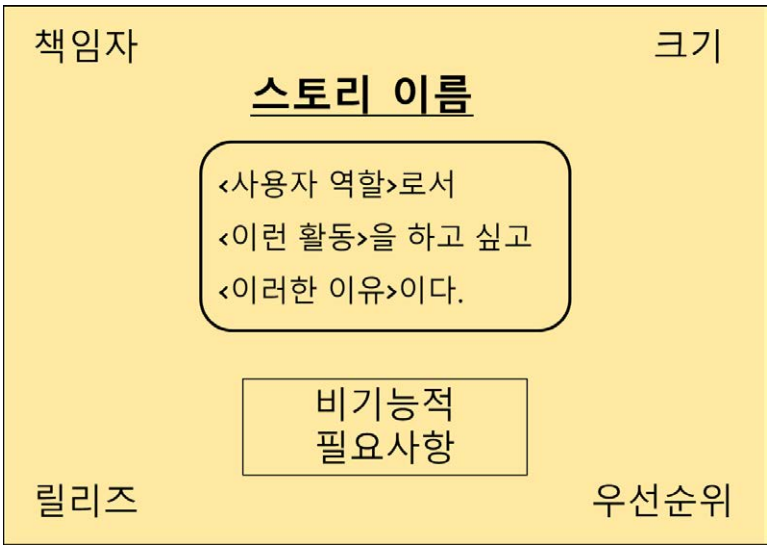
“..로서 (역할)
나는..하기를 원한다(활동)
왜냐하면 (이유)”

전형적으로 사용자 스토리 카드는 다음을 포함한다.

- 스토리와 관련된 비기능적 요구사항들(예를 들어, 보안, 반응 시간 등)
- 책임자 (스토리를 요구했으며 세부사항을 제공할 수 있는 이해관계자)
- 우선순위 (제품 책임자에 의해 결정)
- 추정된 크기 (스토리 포인트)
- 릴리즈
- 인수 기준(카드 뒷면)
- 의존성 (카드 뒷면)

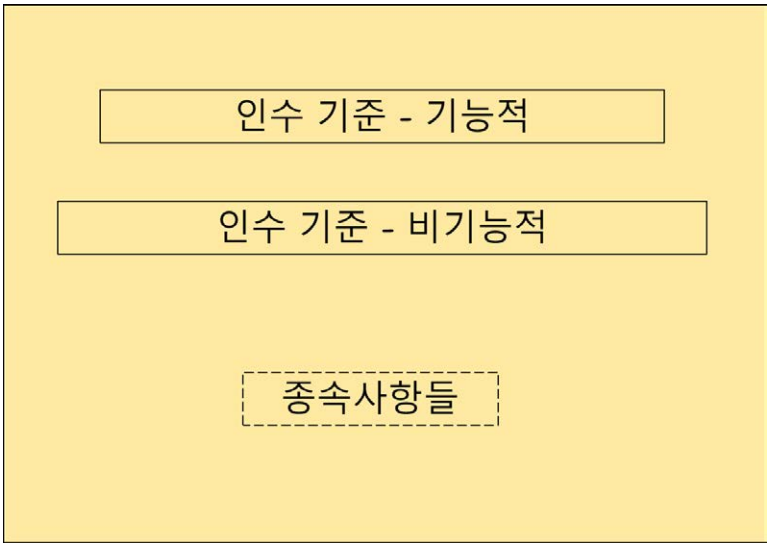
스토리 카드 배치는 아래와 같다. 사용자 스토리들은 보통 A6 카드에 기록된다.

A6 사용자 스토리 카드의 앞면



〈스크럼 개발 프로세스〉

A6 사용자 스토리 카드 뒷면



〈 산출물2 전형적 사용자 스토리 카드 〉

예제 39. 완료된 스토리 카드 앞면

'마이트래블(MyTravel)' 사용자 스토리 로열티 카드 호텔 찾아보기
사용자: 여진
'마이트래블(MyTravel)' 로열티 카드 보유자로서

나는
예약 하기 전에 나의 고객 카드로 호텔 예약 할인 받는 것을 보기 원한다.
왜냐하면
지불한 돈의 가치를 보고 싶고, 잘 대우받는 것을 느끼고 싶기 때문이다

우선순위: 2 크기: 2 릴리즈: 1

예제 40. 완료된 스토리 카드 뒷면

기능 인수 기준:
주어진 고객 카드 로그인이 정확하면 웹페이지에서 "당신에게 제공한다"는 문구가 나타나며 인터넷 접속 기록에 기반한 제공 목록이 나옴.
주어진 고객 카드 로그인이 맞지 않으면 잊어버린 이름과 패스워드에 대한 커멘드 창 표시.
비기능 인수 기준:
보안: 로그인이 안되면, 고객 카드와 개인정보가 가려진다.
성능: TPAC1.2에서 부하와 프로파일 응답을 만족하는 반응 시간 측정
테스트:
스토리 카드 세트:
TPAC1, TPAC2, RTP1, 2, 3
의존성:
사용자 로그인 스토리, 사용자 예약 스토리

6.6.2 사용자 스토리 체크리스트

사용자 스토리를 잘 작성하기 위해 체크리스트를 사용한다. 그리고 체크리스트도 시간이 지남에 따라 리뷰 하고 업데이트해야 한다.

가치	고객, 사용자, 팀에게 가치를 줄 수 있는 기능이 기술돼 있는가?
	우선순위가 기록되었는가?
	협상 가능한가 (예/아니오)?
식별	책임자가 있는가?
	고유하고 짧으며 기능을 설명할 수 있는 이름인가?
내용	포맷이 맞는가?
	설계와 구현이 아니고, 요구사항만 기술하고 있는가?
	제품 책임자와 팀이 이해할 수 있게 기술됐는가?
	사용자와 의미 있는 대화를 할 만큼 기술됐는가?
	비기능 요구사항과 인수 기준을 포함하고 있는가?
	인수 기준이 구체적이고 측정 가능하고 달성 가능하며 현실적이고 테스트 할 수 있는가? (예/아니오)
의존성	의존성이 있는가 (예/아니오)?
	의존성이 식별됐는가?
크기	팀이 이터레이션 계획에서 추정할 만큼 충분한 정보가 있는가?
	추정치가 있는가?
	한 이터레이션 안에 적합할 만큼 작으며 합리적인 크기인가?

〈 산출물3. 사용자 스토리 체크리스트 〉

6.6.3 사용자 스토리 쓰기

스토리는 사용자 요구에 기반하며 종종 각 요구상항은 요구되는 예시 행동을 보여주면서 몇 개의 사용자 스토리로 쓰여진다.

스토리 책임자와 대화가 시작되면 스토리는 빠르고 간결하게 써야 한다.

각 스토리는 한 이터레이션에 적합하도록 작게 할 필요가 있다.

6.6.4 에픽

어떤 사용자 요구사항은 한 이터레이션 내에 완료하기에는 너무 크다. 이것을 에픽이라 불린다. 에픽은 반드시 한 이터레이션 내에 전달될 수 있는 더 작은 사용자 스토리로 나눠져야 한다. 에픽은 복잡한 문제나 광범위한 기술적 변경일 수 있다.
때때로 에픽은 개발이 여러 이터레이션에 걸쳐 연결된 사용자 스토리로 함께 배포될 수 있다. 스토리 간의 상호연결 수준에 따라, 배포 전에 특별한 인수 테스트 이터레이션을 수행할 수 있다.

6.6.5 기술적 사용자 스토리

팀은 직접적으로 사용자 기능에는 해당되지 않지만 개발과 테스팅을 지원해주는 기술적 스토리를 요구할 수 있다. 이런 활동들은 다음과 같다.

- 테스트 도구들
- 인프라 변경
- SDK 업데이트
- "기술 부채" 명확화

팀은 이 스토리를 추정치, 우선순위 등이 준비된 사용자 스토리 포맷에 쓴다. 이 스토리는 제품 백로그에 추가되도록 제품 책임자에게 주어지며, 이터레이션 계획 동안 이터레이션 백로그 안에서 선택될 수 있다.

6.6.6 결함 사용자 스토리

스토리 테스트나 인수 테스트 동안 제기된 결함이 이터레이션 동안 수정되지 않으면, 제품 백로그에 결함 사용자 스토리가 추가된다.

6.6.7 사용자 스토리의 다른 유형

소프트웨어 배포 작업 동안에, 소프트웨어가 사용 가능해지기 전에 다른 사항들이 준비될 필요가 있다. 예를 들어, COTS 소프트웨어를 생산하고 있다면, 고객과 사용자는 아래를 필요로 할 것이다.

- 제품 다운로드 및 설치
- 사용자 가이드
- 업데이트된 마케팅 자료
- 교육 자료

이런 사항들은 팀 밖에서 수행되지만, 소프트웨어 의존성이 있거나 팀이 정보를 제공해야 한다면, 그런 작업은 사용자 스토리가 필요할 수 있다.

예제 41. 소프트웨어 기능이 아닌 사용자 스토리

사용자 외의 사용자 스토리

"교육자로서
나는 시스템 기능의 스크린샷과 설명을 원한다.
왜냐하면 교육 자료로 사용할 필요가 있기 때문이다"

6.7 인수 기준

인수 기준

■ 역할

사용자 스토리의 구현을 확인하기 위한 기준 정의하기

■ 설명

인수 기준은 사용자 스토리의 일부로서 인수 테스팅 동안 사용되는 인수 테스트 세트를 작성하기 위한 기초로 사용된다. 사용자 스토리에 대한 사용자 관점의 세부 사항을 추가하며 종종 정량적 세부 사항 및 설명 옵션을 제공한다.
인수 기준은 다양한 이해 관계자가 작성하며 제품 책임자가 담당한다.

■ 활동

- 제품 백로그 관리
- 인수 테스팅

■ 작업 산출물

- 제품 책임자
- 사용자
- 팀

목적	스토리 완성을 위한 기준 제공
역할과 책임	사용자와 팀의 의견을 반영하는 제품 책임자
보관 / 게시	인수 기준은 A5 사용자 스토리 카드 뒷면에 기록된다.
활동	인수 기준은 제품 백로그 관리의 일부로서 사용자 스토리와 함께, 늦어도 이터레이션 계획 기간 동안, 추정 전에 작성돼야 한다.

〈 표 14. 인수 기준 요약 〉

인수 기준은 중요하다. 전달된 소프트웨어가 인수 가능한지에 대한 합의이다.
인수 기준은 사용자 스토리에 세부사항을 추가하는 데 사용되며, 인수 테스트가 시작되기 전에도 개발자와 테스터에게 유용하며, 인수 테스트의 통과 여부를 위해 사용된다.
사용자 스토리와 밀접하게 관련돼 있으며 스토리 카드의 뒷면에 기록된다.

6.7.1 인수 기준 포맷

인수 기준은 SMART (Specific, Measurable, Achievable, Realistic, Testable, 구체적, 측정 가능, 달성 가능, 현실적, 테스트 가능)라는 면에서 사용자 스토리와 비슷한 특징으로 작성된다. 때때로 비형식적이기 때문에 프로젝트에 적절한 형식을 사용하면 된다.

예제 42. 인수 기준 예시

형식적

네트워크 환경설정의 주어진 작업량 하에, 시스템을 가동하면서 성능 테스트 세트 12를 사용해 측정되는 반응 시간은 평균적으로 1.5초, 항상 3초 이내여야 한다.

Given-Then 포맷:

Given네트워크 환경설정의 주어진 작업량 하에, 시스템을 가동하면서 성능 테스트 세트12를 사용한다.
Then 반응 시간은 평균적으로 1.5초, 항상 3초 이내여야 한다.

비공식적

새로운 기능은 훈련 코스 없이도 반드시 사용하기 쉬워야 한다. 사용자가 화면의 프롬프트와 팝업의 추가적 도움의 필요 없이 기능을 이해하고 사용할 수 있기를 기대한다.

Given-Then 포맷:

Given훈련이나 추가적 도움이 주어지지 않는다.
Then 사용자는 새로운 기능을 이해하고 사용할 수 있어야 한다.
Given 사용자가 마우스를 버튼에 가져간다.
Then 관련 도움 팝업 창이 나타난다.

보여진 대로, 인수기준은 형식적, 비형식적으로 쓸 수 있다. 만일 'Given-Then' 포맷을 사용한다면, 테스트 자동화를 할 수 있다.

Tip 46. 왜 인수 기준이 형식적이어야 하는가?

짝 프로그래밍에서는 두 프로그래머가 하나의 컴퓨터를 공유한다. 이는 새로운 팀원을 참여시키거나 결함 해결 시 논의할 수 있어서 좋다. 짝 프로그래밍 방법으로 코드의 품질을 향상시킬 수 있다.

6.7.2 인수 기준 수립

제품 책임자는 팀과 함께 사용자와 이야기하고 인수 기준을 명확히 하기 위해 작업한다. 다음을 고려해야 한다.

- 인수 기준은 '인수가 가능하냐?'에 대한 것이지 '완벽'을 위한 것이 아니다.
- 인수 기준은 반드시 SMART (Specific, Measurable, Achievable, Realistic, Testable)해야 한다. 구체적, 측정 가능, 달성 가능, 현실적, 테스트 가능 해야 한다.
- 인수 기준은 기능, 비기능 요구사항을 포함한다.
- 사용자가 무엇을 원하는지, 무엇을 요구하고 있는지를 완전히 이해하지 못할 수도 있다.
- 사용자가 비기능 속성에 필요한 것을 이해하기 어렵다.

인수 기준에 대해 사용자에게 간접적으로 질문하는 것이 때때로 유용하다.

Tip 47. 인수 기준에 대해 질문하기

예를 들어, 신뢰성 수준에 대해 알고 싶다면, "소프트웨어는 얼마나 신뢰성이 있어야 하는가?"라고 질문하기 보다는 "소프트웨어 없이 얼마나 오래 작업할 수 있는가?" 또는 "만일 소프트웨어가 한 시간 동안 작동되지 않으면 얼마나 많은 돈을 잃겠는가?" 라고 질문하는 것이 유용하다.

사용자가 프로젝트의 예산 및 일정으로 현실적이지 못한 인수 기준을 요구한다면, 더 현실적인 인수 기준을 선택할 수 있는 옵션을 제공한다.

예제 43. 인수 기준 협의

'마이트래블'(MyTravel) 프로젝트: 인수 기준 논하기

'마이트래블'(MyTravel) 팀 리더인 '요한'과 '마이트래블' 제품 책임자인 '숙자'는 소프트웨어 사용자 중 한 명인 '여진'을 만난다. 그들은 다음 릴리즈의 사용자 스토리 인수 기준을 수립하기 위해 한 시간으로 제한된 워크숍을 갖는다

'숙자'는 다음 릴리즈 목표, 사용자 스토리 제목의 목록을 갖고 있으며, 시간 규모가 제한돼 있음을 알고 있다.

'숙자'가 말한다. "이번 릴리즈는 '마이트래블(MyTravel)' 프런트 오피스 시스템 업데이트를 지원하는 것인데 요약하면, 성능 업그레이드가 새로운 기능 구현보다 당신에게 더 중요한 것으로 보입니다. 맞지요?"

'여진'이 말한다. "맞아요. 작업의 대부분을 할 수 있지만, 시스템이 느리며 때때로 거쳐야 할 단계가 너무 많아요. 모든 동작이 1초 이내의 반응 시간이 필요합니다."

'요한'이 말한다. "음, 모든 동작에서 1초 이내의 반응 시간은 가능하지만, 시스템을 다시 설계해야 하기 때문에 비용이 비싸고 기간이 많이 소요될 수 있어요. 이게 바른 방법인지는 생각해 볼 필요가 있어요."

그들 셋은 업무 및 무엇이 오랜 시간이 걸릴지에 대해 이야기하면서 약간의 시간을 보낸다.

"그럼, 만약 자주 사용하는 업무를 완성하는 데에 더 적은 단계를 거칠 수 있다면, 일정을 앞당길 수 있나요?" '숙자'가 묻는다. '여진'은 동의한다. "그렇게 되면 일정을 앞당길 수 있을 겁니다."

'요한'이 제안한다. "가장 자주 사용하는 업무를 위한 인터페이스와 사용자 경로의 일부를 검토해보고 재설계를 해보는 게 어떨까요? 그리고 느리지만 중요하며 자주 사용되는 중요한 세 부분을 성능 튜닝을 하는 것입니다."

'숙자'와 '여진'은 그 방안이 효과가 있을 것이며, 시간과 예산 제한 하에서 개선이 이루어질 것에 동의한다.

그들은 Given-Then 포맷으로 중요한 세 부분에 대한 성능 인수 기준에 동의한다.

Given 레퍼런스 참조해 성능 테스팅을 수행한다.

Then 반응 시간은 반드시 항상 3초 이내여야 하고 평균적으로 1.5초 이내여야 한다.

그리고 그들은 가장 빈번한 업무에 대해 비형식적 사용자 인터페이스 및 경로를 인수 기준으로 동의한다.

비형식적: 가장 빈번한 업무는 반드시 화면 변경 4번 이내 및 20번의 키보드 입력 또는 클릭 이내여야 한다.

6.7.3 인수 기준 측정

테스트 동안 인수 기준을 측정한다. 인수 기준은 제품 책임자와 사용자에게 스토리가 완료되었음을 보여주는 데 사용된다. 스토리 테스트 동안 인수 기준을 측정하는 것은 유용하다.

팀의 개발자나 테스터는 아래와 같은 인수 기준을 사용할 수 있다.

- 사용자 요구사항을 이해했는지 확인
- 스토리 코드 개발
- 단위, 스토리, 인수 테스트 설계

소프트웨어가 요구된 기능과 비기능적 요구사항을 만족하는지 측정하는 데 인수 기준을 사용한다.

예제 44. 인수 기준 충족하기

인수 기준을 사용해 작업하는 '마이트래블(MyTravel)'팀

'마이트래블(MyTravel)' 팀 리더인 '요한'과 제품 책임자인 '숙자'는 UX 디자이너인 '혜림'과 개발자인 '치원', 테스터인 '철순'과 이야기한다. 그들은 인수 기준에 대해 토론한다.
성능 인수 기준:
Given 레퍼런스 참조해 성능 테스팅을 수행한다.
Then 반응 시간은 반드시 항상 3초 이내여야 하고 평균적으로 1.5초 이내여야 한다
가장 빈번한 업무를 위한 비격식적 사용자 인터페이스 및 경로 인수 기준: 비격식적: 이것들(특정 업무들)은 화면 변경 4번 이내 및 20번의 키보드 입력 또는 클릭 이내에 완성 가능해야 한다.
비형식적 가장 빈번한 업무는 반드시 화면 변경 4번 이내 및 20번의 키보드 입력 또는 클릭 이내여야 한다.
'혜림'은 사용자 인터페이스 인수 기준을 채택하며, 사용자 인터페이스를 리뷰 한다. 그녀는 새로운 인터페이스를 설계하며, '철순'과 함께 이를 리뷰 한다.
'치원'은 성능 인수 기준을 채택하며, 성능을 최적화하고 튜닝 하는 코드를 리뷰 한다.
'치원'과 '요한'은 새로운 인터페이스를 개발하고 성능 튜닝을 수행한다.
'철순'과 '혜림'은 예상 결과가 인수 기준에 맞도록 가장 빈번하고 중요한 사용자 작업에 대한 테스트를 잘 하기 위해 방안을 생각한다.

6.8 작업상황판

작업상황판

■ 목적

이터레이션 동안 진행 상황의 검토 수단 제공

■ 설명

작업상황판은 이터레이션 동안 사용자 스토리의 진행 상황을 기록하기 위한 가시적이고 구조화된 수단을 제공한다. 작업상황판은, 상황 또는 활동을 나타내는, 각 사용자 스토리가 왼쪽에서 오른쪽으로 진행돼야 하는 여러 개의 열(columns)로 구성된다.
활동(예: 설계 및 코딩, 스토리 테스팅)를 완료할 때 팀원이 작업상황판을 가로질러 사용자 스토리 카드를 이동시키지만 전체적인 작업상황판은 팀 리더가 담당한다.

■ 활동

- 이터레이션 계획
- 개발 및 테스팅
- 인수 테스팅
- 일일 스탠드업 미팅

■ 역할

- 팀 리더
- 팀
- 제품 책임자

목적	수행돼야 할 작업과 이터레이션의 현재 진행 상황에 대해 팀과 이해관계자들에게 시각적으로 공유하는 것
역할과 책임	팀원은 사용자 스토리 카드를 이동시켜서 스토리의 진행 상황을 반영함으로써 작업상황판을 업데이트 한다. 팀 리더는 작업상황판의 계속적인 업데이트를 확인할 책임이 있다.
보관 / 게시	작업상황판은 팀 사무실의 벽면에 게시돼 볼 수 있어야 한다. 작업상황판은 화이트보드일 수 있지만 일반적으로 게시판 위나 벽면에 설치되는 보다 영구적인 물품일 수 있다.
활동	각 사용자 스토리는, 설계, 코딩, 스토리 테스팅, 인수 테스팅, 배포와 같은 일련의 작업을 거친다. 작업상황판 위에는 이터레이션 백로그를 위한 하나의 열이 있고, 각 작업 유형을 위한 하나씩의 열이 있다. 사용자 스토리 카드는 각 작업이 완료될 때 작업상황판을 가로 질러 이동된다.

〈 표 15. 작업상황판 요약 〉

작업 현황판은 세로줄로 나뉘는 화이트보드 또는 게시판이다. 첫 번째 세로줄은 “이터레이션 백로그”로 분류된다. 다른 세로줄은 개발, 테스팅 또는 배포 활동 등을 나타낸다. 스토리 카드는 진행됨에 따라 이터레이션 백로그로부터 작업 현황판에 걸쳐 이동한다. 따라서 작업 현황판은 이터레이션에서의 진행상황을 잘 보여준다.
사용자 스토리 카드는 처음에 이터레이션 백로그 세로줄의 작업 현황판에 있다.
개발자가 스토리의 설계와 코딩을 시작하기로 결심했을 때 스토리 카드를 ‘개발’으로 옮긴다. 스토리 완료 정의가 달성되면, 카드는 ‘스토리 테스트’으로 옮겨진다.
테스터는 스토리 테스트가 완료되면 ‘인수 테스트’로 옮긴다. 인수 테스트를 통과한 스토리는 ‘배포’로 옮긴다.
‘배포’ 후에는 스토리 카드를 다음 줄로 이동시켜 완전히 완료시킨다.
스토리 카드는 거꾸로 이동할 수 있는데 결함이 테스트에서 발견될 때 일어난다. 테스터는 반드시 스토리를 수행하는 개발자에게 재 작업에 대해 알려야 한다
누가 작업하고 있는지 알기 위해 이름, 사진, 아바타를 스토리 카드에 추가하는 것이 유용할 수 있다.

작업 현황판 예시

PIP 프로젝트를 위한 이터레이션 10진행

예제 45. 이터레이션 진행중 작업 현황판

PIP 이터레이션10:
스토리 10개. 예상 속도: 32 스토리 포인트
재규 얼른 회복하기를-그립다!

이터레이션 백로그	개발	스토리 테스트	인수 테스트	배포
S7 ☐ S9 ☐	S8 ☐	S1 ☐ S2 ☐ S3 ☐ S6 ☐	S5 ☐ S10 ☐	S4 ☐

6.9 테스트 차터

테스트 차터는 탐색적 테스트를 수행할 때 사용된다. 테스트 차터는 스토리의 리스크와 범위에 대해 생각하고 토론함으로써 만들어진다.
세션 노트가 있는 테스트 차터 템플릿은 다음과 같다.

<table>
<tr><td>테스트 차터 번호:</td><td>프로젝트 이름:</td><td>이터레이션 번호:</td><td>사용자 스토리 참조:</td></tr>
<tr><td>개발자:</td><td>테스터:</td><td colspan="2">사용자:</td></tr>
<tr><td colspan="4">탐색한 내용:

사용한 도구:

발견한 내용:</td></tr>
<tr><td colspan="4">리스크:</td></tr>
<tr><td colspan="4">타임박스:</td></tr>
<tr><td colspan="4">세션 노트</td></tr>
<tr><td colspan="2">단 계</td><td colspan="2">발견사항</td></tr>
<tr><td colspan="2"></td><td colspan="2"></td></tr>
</table>

〈 산출물4. 테스트 차터 템플릿 〉

6.10 인수 테스트 세트

인수 테스트 세트

■ 목적

사용자 스토리가 올바르게 구현되었는지 확인

■ 설명

사용자 스토리를 위한 인수 테스트 세트는 사용자 스토리 및 관련된 이해 관계자와의 대화를 통해 얻어지는 필요한 행위를 파악해 작성된다.
명세화된 테스트 세트는 인수 테스팅 동안 사용자가 사용할 수 있도록 팀이 작성한다. 이것은 일반적으로 시스템의 상위 리스크 부분과 연관돼 있다. 이 테스트 중 일부는 자동화돼 향후 회귀 테스팅에 사용된다.
인수 테스트 세트는 팀이 담당한다.

■ 활동

- 테스팅

■ 역할

- 팀원
- 사용자
- 제품 책임자

목적	**합의된 인수 기준이 충족되고 구현된 코드가 합의된 사용자 스토리와 일치하는지를 확인하는 인수 테스팅을 사용자가 수행할 수 있도록 돕기 위해 팀은 인수 테스트 세트를 작성한다.**
역할과 책임	**테스팅을 담당하는 팀원이 인수 테스트 세트를 작성한다.**
보관 / 게시	**인수 테스트는 테스트 차터의 형태를 취할 수 있으며 스프레드 시트에 보다 상세하게 명세화 할 수 있다.**
활동	**인수 테스트 세트는 테스팅 활동의 일환으로 이터레이션 동안 작성되며 인수 테스팅을 위해 준비된다.**

〈 표16. 인수 테스트 세트 요약 〉

인수 테스트 세트 차터 번호:	프로젝트 이름:	이터레이션 번호:	사용자 스토리 참조:
개발자:	테스터:	사용자:	
인수 기준: 탐색한 내용: 사용한 도구: 발견한 내용:			
사용자 스토리 기본 흐름 탐색한 내용 사용한 도구: 발견한 내용:			
사용자 스토리 대안 흐름 탐색한 내용: 사용한 도구: 발견한 내용:			
가장 중요한 업무 사용자 스토리에 대해 가장 중요한 작업을 여기에 나열하고, 소프트웨어에서 시도할 때 체크 하시오			

〈 산출물 5. 인수 테스트 세트 테스트 차터 〉

6.11 참조사항

6.11.1 스토리 페르소나

소프트웨어의 사용자와 직접적으로 만날 수 없는 경우가 있다. 공공 웹사이트를 구축하거나, COTS 소프트웨어를 개발하거나, 제3자가 사용할 소프트웨어를 개발한다면, 사용자를 모를 수 있다. 또한 사용자가 너무 바쁘면 만날 수 없다.
이런 상황에서, 사용자 개인의 요구를 이해하기 위해 페르소나를 사용할 수 있다.

Tip 48. 페르소나 사용하기

사용자와 접촉할 수 있어도, 페르소나는 소프트웨어를 왜 구축하는지? 누가 그것을 사용하는지에 대한 초점을 맞추는데 도움을 주기 때문에 많이 사용된다.

페르소나는 시스템의 사용자, 사용하는 환경, 개인적 특성을 기술한다. 페르소나는 사용자의 능력, 동기부여, 방해물, 왜 그들은 그 시스템을 사용하고 싶어하는지를 기술한다.
웹사이트나COTS 소프트웨어를 구축하고 있다면, 마케팅 팀이 페르소나가 제품 기능을 정의하도록 규정했음을 알게 될 것이다. 그러한 경우 이미 만든 페르소나를 재 사용할 수 있다. 또한 판매, 지원, 교육 부분에 있는 사람들로부터 사용자가 원하는 것에 대한 이해를 도움 받을 수 있다.

예제 46. 오키드딜라이트 페르소나 예시 – 철순

'오키드 딜라이트'(OrchidDelights)를 위한 페르소나

'오키드 딜라이트'(OrchidDelights) 웹 프로젝트는 마케팅 팀에 웹사이트를 위한 페르소나를 요구했다. 마케팅 팀이 만든 페르소나 두 가지가 아래에 있다.
오키드 딜라이트'(OrchidDelights 웹사이트 역할 – 구매자 이름 – 철순 
철순은 25살이며 전문적이고 야심이 있다. 그는 25살이며 전문적이고 야심 있는 미란과 약혼했다. 그들은 패션 색깔과 선물 유형에서 유행을 따르며 사치품을 좋아한다. 그들은 친구 집단에서 유행을 이끌고 싶어한다.
철순은 말다툼 후에 미란에게 줄 꽃을 사려고 한다.
철순은 기술 적응력이 높다. 그는 자신의 아이패드와 아이폰을 항시 접속 유지하며 비즈니스 및 여가에 사용한다. 그는 구매할 물품을 빠르게 검색하고 당일배송을 기대한다. 그는 늦게 예약된 당일 배송에 대해 기꺼이 추가 비용을 지불할 것이다.
웹사이트로부터: 선택하고 둘러볼 수 있고, 장바구니에 넣을 수 있고, 그의 애플 기기를 사용해 애플 페이를 통한 원 클릭 구매를 할 수 있기를 기대한다. 웹사이트의 빠른 속도, 보안, 신뢰성은 당연하다. 반응이 0.5초 이상 기다리게 되면 그는 클릭해서 나가버릴 것이다. 그는 새로운 애플 기기가 출시 되자마자 업그레이드 할 것이다. 그는 웹사이트가 열리는 때부터 당일 배송이 보장될 때까지의 전체 구매 과정에 5분 이하가 걸리기를 원한다.
제품으로부터: 꽃 진열을 위한 최신 색깔과 디자인. 이미 꽃병에 정렬돼 있어서 정렬할 필요가 없어야 함. 활짝 핀 상태. 그 밖의 유행 사항들 선택.

예제 47. 오키드딜라이트 페르소나 예시– 광

오키드 딜라이트'(OrchidDelights 웹사이트 역할 – 구매자 이름 – 광 

광은 65세이며 은퇴했다. 그의 딸 혜림의 생일파티가 있다. 광은 전통적이고, 훌륭한 가치를 좋아하며, 사려 깊다. 그는 딸의 생일에 딸에게 줄 꽃을 사고 싶어한다.

광은 온라인 구매에 윈도우 노트북을 사용한다. 그는 브라우저로 인터넷 익스플로러를 사용한다. 그는 뭔가를 할 필요가 있을 때에만 노트북을 사용한다. 그는 인터넷 검색을 할 수 있고, 아이디어를 저장할 수 있고, 나중에 다시 찾을 수 있기를 기대한다. 그는 미리 계획하고 돈을 절약하는 것을 좋아한다. 따라서 그는 며칠 미리 구매해서 배송 비용을 아끼게 되면 매우 기뻐한다.

웹사이트로부터: 선택하고 둘러볼 수 있고, 장바구니에 넣을 수 있고, 다른 날을 위해 구매 리스트를 저장할 수 있기를 기대한다. 신용카드나 현금카드를 사용해 구매할 수 있어야 한다. 웹사이트의 명확한 설명, 보안 그리고 신뢰성이 요구된다. 그는 지불할 때 웹사이트의 그림을 살펴보고 확인하면서 많은 시간을 보내기를 원한다. 따라서 빠르게 시간 제한이 되기를 원치 않는다. 그는 구매가 완료되었고 배송이 완료되었다는 확인 메시지로 안심시켜줄 필요가 있다.

제품으로부터: 꽃 진열을 위한 전통적 색깔과 디자인. 이미 꽃병에 정렬된 상태뿐 아니라 꽃다발 세트로 정렬돼야 함. 가격 대비 좋은 물건. 그 밖의 전통적 선물 선택.

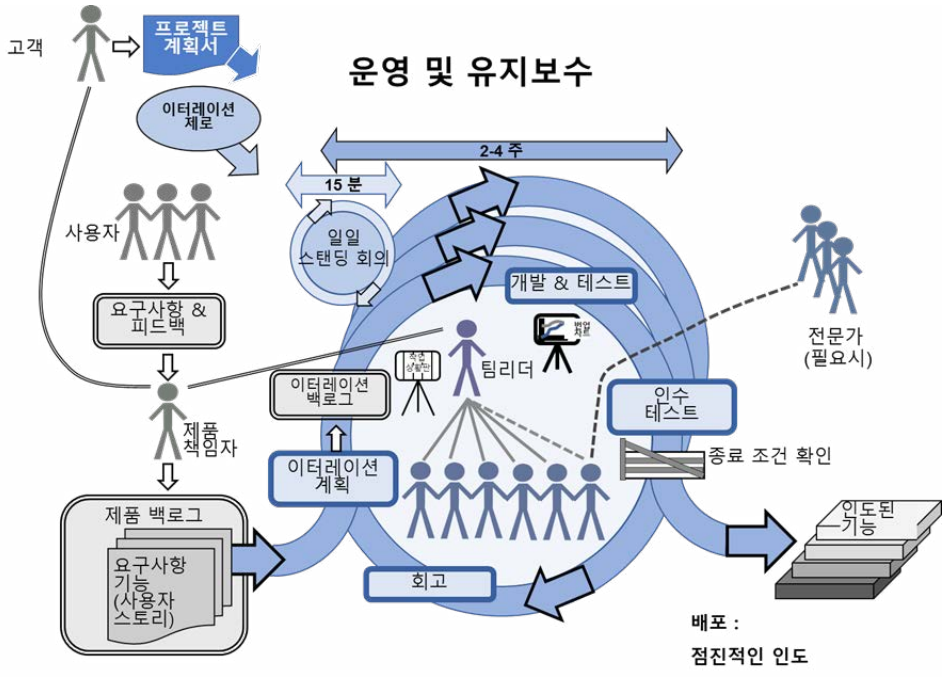
CHAPTER 07

운영 및 유지보수 수행 활동

CHAPTER 07 운영 및 유지보수 수행 활동

7.1 운영 및 유지보수 개요

운영 및 유지보수는 프로젝트 개발이 완료되어 사용자에게 소프트웨어가 릴리즈되면 시작되며 개발된 시스템의 결함을 해결하거나 사용자의 요구를 지속적으로 반영하면서 기능을 개선하는 일들을 수행한다.



〈스크럼 개발 프로세스〉

이 단계에서 발생되는 주요 업무는 다음과 같다.

- 기존 기능에 대한 개선
- 결함에 대한 해결
- 새로운 기능의 추가
- 비기능적인 속성 개선

유지보수를 수행하는 팀에서는 신규 기능을 개발하는 것보다 기존 기능에 대한 개선요청이나 급하게 요청하는 결함에 대한 해결 작업이 많아진다. 따라서 개발단계와는 업무 프로세스가 많이 달라진다. 본 가이드를 유지보수 조직에 적용할 때는 릴리즈 계획 같은 활동은 필요 없으나 제품백로그 관리나 이터레이션 계획, 일일 스탠드업 미팅, 회고, 지속적 통합과 같은 활동은 그대로 적용된다.

7.2 업무 흐름에 대한 시각화

유지보수 업무를 수행하는 팀에서는 고객의 요구에 빠르게 대응하면서 품질을 확보하는 것이 중요하므로 고객 요구에서 배포까지의 수행되는 업무 흐름을 먼저 이해하고 이를 시각화하는 것이 필요하다. 유지보수 조직은 기존 기능에 대한 개선요청이나 결함에 대한 해결 작업이 많기 때문에 두 가지 경우를 고려하여 작업 현황판(Task Board)을 설계한다. 다음은 기능 개선과 결함 해결에 대한 일반적인 업무 흐름도이다.

〈 도표 17. 기능개선 업무 프로세스 〉

〈 도표 18. 결함 해결 업무 프로세스 〉

상기 2가지 업무흐름을 바탕으로 아래와 같이 작업 상황판을 구축할 수 있다. 현황판의 첫 번째 '제품백로그' 열은 내부적으로 요구사항에 대한 검토가 끝나고 서비스 개선 기획이 완료된 요구기능(스토리)들이 담긴다. 이 때 상단에 있는 것이 우선순위가 높다. 두 번째 '긴급'열은 비 정기적으로 발생되는 결함이나 경영진의 급한 요구들이 담기는 곳이다. 개발자들은 본인 작업이 끝나면 '제품백로그' 에 있는 스토리보다 결함을 우선적으로 가져간다. 세 번째 '개발'열은 개발자들이 결함과 스토리를 가져와 개발을 진행하는 곳이다. 개발자들이 개발을 완료하면 해당 스토리는 개발완료로 이동한다. 테스터는 개발 완료된 스토리를 가져와 테스트를 수행한 후 '테스트 완료' 열로 이동시킨다. 인수테스팅은 제품책임자와 사용자가 중심이 되어 수행한다.

제품 백로그	긴급	개발	개발 완료	테스팅	테스트 완료	인수 테스팅	인수테스트 완료	배포 완료
S17 ☐ S18 ☐ S19 ☐ S20 ☐	D10 ☐ D11 ☐ S16 ☐	S14 ☐ S15 ☐ D9 ☐	S12 ☐ S13 ☐ D7 ☐ D8 ☐	S10 ☐ S11 ☐ D5 ☐ D6 ☐	S8 ☐ S9 ☐ D3 ☐ D4 ☐	S5 ☐ S7 ☐ S6 ☐	S3 ☐ S4 ☐	S1 ☐ S2 ☐ D1 ☐ D2 ☐

〈 도표 19. 유지보수 작업 진행 상황판 사례 1 〉

작업진행 상황판은 상기 형태 외에도 다양하게 작성할 수 있는데 아래는 첫 번째 열에 팀에서 수행하는 서비스 유형을 구분하였다.

구 분	제품 백로그	개발	개발 완료	테스팅	테스트 완료	인수 테스팅	인수테스트 완료	배포 완료
긴급	D9 □ S20 □	D8 □	D7	D5 □ D6 □	D3 □ D4 □	S6 □		D1 □ D2 □
제품X	S17 □ S18 □	S14 □ S15 □	S12 □ S13 □	S10 □	S8 □ S9 □	S5 □ S7 □	S3 □ S4 □	S1 □ S2 □
제품 Y	S17 □ S18	S14 □ S15 □	S12 □ S13 □	S10 □	S8 □ S9 □	S5 □ S7 □	S3 □ S4 □	S1 □ S2 □

〈 도표 20. 유지보수 작업 진행 상황판 사례 2 〉

상황판이 완성되면 개발팀에서 현재 진행하고 있는 업무를 포스트잇을 활용하여 상황판에 붙인다. (서비스 유형별로 포스트잇 색깔을 다르게 하여 구별하는 것이 좋다.) 포스트잇의 상단에는 스토리포인트(SP)나 투입공수(MD)를 기록한다. 개발자들의 모든 업무가 상황판에 붙게 되면 누가 어떤 일을 하고 있으며 어떤 공정에 일이 많이 몰려있는지 쉽게 알 수가 있다.

S7 3 MD
NFC/ RFID 카드 등록 화면 추가
(why) 카드 등록 시 사용자 확인 필요

〈 도표 21. 유지보수 스토리 카드 샘플 〉

7.3 제품백로그 관리

유지보수 팀에서도 제품 백로그 관리는 필요하다. 제품책임자는 다양한 사용자로부터 나오는 개선 요구사항을 검토해서 제품에 가치 있는 사용자스토리로 변환시키고 우선순위를 선정한다.

7.3.1 요구사항 검토

제품책임자는 사용자로부터 들어오는 개선 요구사항과 경영진, 영업 등으로부터 제안되는 요구사항 등을 검토하여 상충되는 요구사항이나 불명확한 부분이 있는지 점검하고 제품의 가치를 높일 수 있는 요구사항을 식별한다. 식별된 요구사항은 사용자 스토리 형태로 작성하고 이해관계자들과 협의하여 개발 진행 여부를 확인한다. 필요시 기획서 및 화면디자인 작업을 수행한다.

7.3.2 제품백로그 우선순위 미팅

제품책임자는 개발하기로 확정된 스토리들의 우선순위 선정을 위하여 경영진 및 이해관계자들과 주기적인 미팅을 수행한다. 이 과정에서 다음과 같은 작업이 수행된다.

- 스토리 우선순위 조정
- 더 이상 필요 없어진 스토리의 제거
- 오래된 스토리에 대한 재검토

사용자 및 영업팀 등에서 들어오는 다양한 서비스 요청들을 검토해서 가치 있는 것들을 식별하면 이것들이 아래 그림의 첫 번째 '서비스요청' 열에 담긴다. 이중에서 기획서나 디자인 문서작업이 필요한 것은 '기획 중'열로 이동한다. 기획이 완료되고 최종적으로 경영진이 승인한 사항은 개발승인으로 이동한다. '개발승인' 열에는 우선순위를 설정한다. 우선순위는 사용자의 요청일자 및 비즈니스 가치 측면을 고려하여 먼저 개발할 것을 가장 상단에 놓고 하단으로 내려갈수록 나중에 개발 할 것을 놓는다.

서비스 요청	기획 중	기획완료	개발승인
서비스 요구 □ 서비스 요구 □ 서비스 요구 □ 서비스 요구 □	서비스 요구 □ 서비스 요구 □	서비스 요구 □ 서비스 요구 □	서비스 요구 □ 서비스 요구 □

우선순위

〈 도표 22. 기획 작업 상황판 사례 〉

7.3.3 제품백로그 보급 방법

제품백로그에 대한 보충은 다음과 같은 2가지 방식으로 보급할 수 있다.

■ **이터레이션 계획 수립**

- 일반적으로 개발팀에서는 기존 기능을 개선하거나 신규로 개발해야 할 일들이 쌓여 있는 경우가 많다. 이럴 경우 주기적인 이터레이션 계획 수립 활동을 통하여 작업 진행 상황판에 있는 '제품백로그' 열에 스토리들을 보급한다.
- 개발팀에서는 백로그 우선순위에 따라 해당 이터레이션에서 수행할 수 있는 양만큼의 스토리들을 가져오고 팀원들에게 할당한다. (6.5 이터레이션 계획 참조)

■ **진행 중 업무 개수에 대한 제한**

- 작업 진행 상황판에 있는 '제품백로그' 열에 들어갈 수 있는 스토리의 개수를 제한하고 해당 개수가 비워질 때만 스토리를 보급하는 방식이다. 개발자들은 자기 일이 끝나면 제품백로그에서 우선순위가 높은 스토리들을 자발적으로 가져간다.
- 예를 들어 '제품백로그' 열에 들어갈 수 있는 스토리의 개수를 10으로 제한하면 해당 열에는 10개의 스토리만 존재해야 한다. 만약 추가로 스토리 2개를 넣고자 한다면 개발팀에서 2개를 가져갈 때까지 기다려야 한다.
- 이 방식은 이터레이션 방식과는 다르게 사전에 스토리들이 할당되지 않으므로 팀 리더는 언제라도 요구사항 우선순위를 조정 할 수가 있다. 따라서 급하게 요구하는 사항들이 많은 팀에 적절하다.

제품백로그 (10)	결함	개발 중	개발완료	테스트 중	테스트 완료	배포완료
(S1□, S2 □), S3 □, S4 □ S5 □, S6 □ S7 □, S8 □ S9 □, S10 □	D1 □ D2 □	S1 □, S2 □				

S10□, S11 □

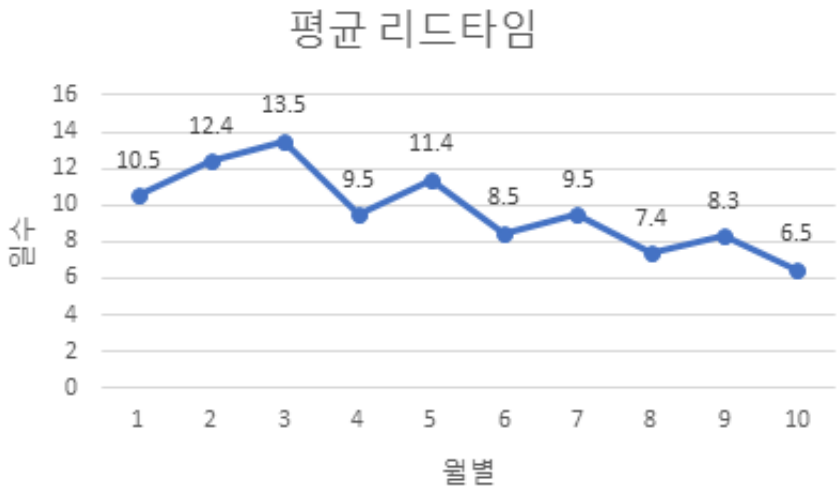
〈 도표 23. 제품 백로그 보급방법 〉

7.4 업무 흐름 관리

7.4.1 버틀넥 해소

많은 개발팀에서 테스터가 부족하다 보니 개발은 완료되었으나 테스팅에 버틀넥이 걸려있는 경우가 많다. 이렇게 되면 고객에게 전달되는 리드타임(Lead Time)이 길어질 수 밖에 없다. 팀 리더는 이를 방지하기 위하여 작업 진행 상황판을 매일 점검하고 버틀넥이 걸리는 곳이 어디인지 파악해야 한다. 버틀넥이 생기는 공정을 발견하면 팀원 전체가 협력하여 이것을 지속적으로 해소시켜 주어야 한다.

- 리드타임(Lead Time) 은 고객 요청에서 배포까지 걸리는 총 시간을 측정한 값으로 아래 그림과 같이 표현될 수 있다. 평균 리드타임은 월별로 수행한 모든 스토리들의 리드타임을 평균한 값이다. 아래 그림을 보면 처음에는 다소 증가 추세를 보이다가 버틀넥을 지속적으로 해소하면서 평균 리드타임이 감소하는 것을 볼 수 있다.



〈 도표 24. 스토리 평균 리드타임(Average Lead Time)〉

7.4.2 일일 스탠드업 미팅

유지보수 팀에서도 일일 스탠드 업 미팅은 가급적 매일 15~20분 정도 수행한다. 이때 테스터 및 제품책임자, 디자이너 등이 모두 참여하는 것이 좋다. 담당자는 어제 한 일과 장애요인, 금일 할일, 도움이 필요한 사항 등을 1분 내외로 얘기한다. 진행방법은 5.5 일일 스탠드업 미팅을 참조한다.

7.4.3 주기적인 회고

유지보수 팀에서 회고는 이터레이션 말이나 정기적으로 2~4주 단위로 수행한다. 진행방법은 4.11 회고 미팅을 참조한다.

CHAPTER 08

시범적용 사례

CHAPTER 08

시범적용 사례

8.1 W사 사례

8.1.1 조직 개요

이 회사는 체육학원 (체력측정 및 출결 알림 시스템 등) 및 일반 학원(출결, 수납, 셔틀버스 위치관제 등) 관리 프로그램, 식생활 개선 프로그램 등을 개발하는 전문 솔루션업체이다. 전체 직원은 20여명 정도로 영업, 기획, 개발팀 등으로 구성되어 있으며 개발팀은 웹 개발자 2명, 앱 개발자 4명(안드로이드, IOS), 디자이너 1명, 테스터 1명으로 총 7명으로 구성되어 있다. 기획자 2명은 개발팀과 밀접하게 일하고 있으며 전체 기획과 개발팀을 이끌고 있는 팀장이 1명 있다.

8.1.2 기존 업무 현황 및 프로세스

이 조직은 몇 개의 솔루션들을 가지고 있으며 현재는 이를 개선하고 유지하는 일을 주로 수행한다. 주요 업무는 아래와 같다.

- 기존 솔루션에 대한 기능 추가 및 개선
- 기존 솔루션에 대한 문의 및 장애 응대
- 결함 해결

업무진행은 아래와 같은 절차로 진행된다.

1) 서비스 기획

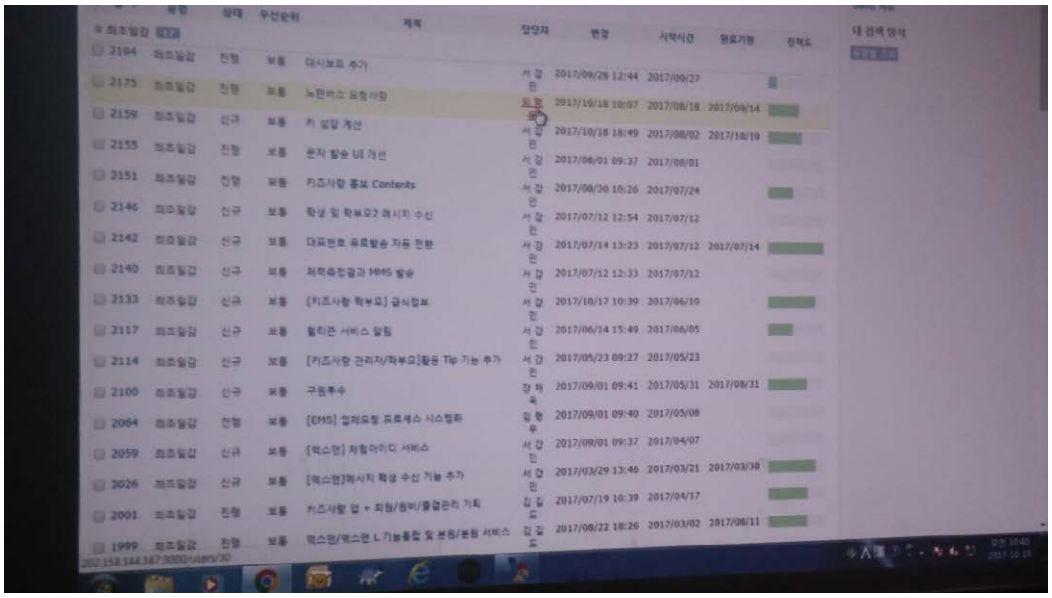
- 현장 사용자 요구를 바탕으로 영업과 경영진이 새로운 서비스나 기능 개선을 요청하면 기획자는 경영진과 밀접하게 소통하면서 기획서를 작성한다.
- 완료된 기획서는 레드마인(Redmine) 도구에 일감으로 등록된다.

2) 디자인 및 개발

- 완성된 기획서를 바탕으로 디자인 작업이 먼저 수행되며 디자인이 필요 없는 작업은 개발팀에서 업무를 수행한다. 업무할당은 팀장 주도로 디자인 및 안드로이드, IOS, 웹 개발 등을 맡은 각각의 담당자에게 분배된다.
- 팀원들은 맡은 작업이 완료되면 팀장에게 구두로 알리고 레드마인(Redmine)에 기록한다.

3)테스트

- 테스터는 개발자가 완료한 기능을 테스트하며 발견된 결함은 개발자에게 전달된다.
- 기획자는 테스팅이 끝난 작업에 대해서 인수테스팅을 수행하지만 현재는 테스팅 업무가 많아서 기획자만 테스팅을 수행하고 배포하는 경우가 많음



〈 도표 25. 레드마인에 등록된 일감 목록 〉

8.1.3 주요 문제점

이 곳은 작은 조직임에도 불구하고 구성원들간에 직접적인 소통이 부족하고 팀장이 모든 것을 챙기고 지시해야 하는 상황으로 다음과 같은 문제점들이 있었다.

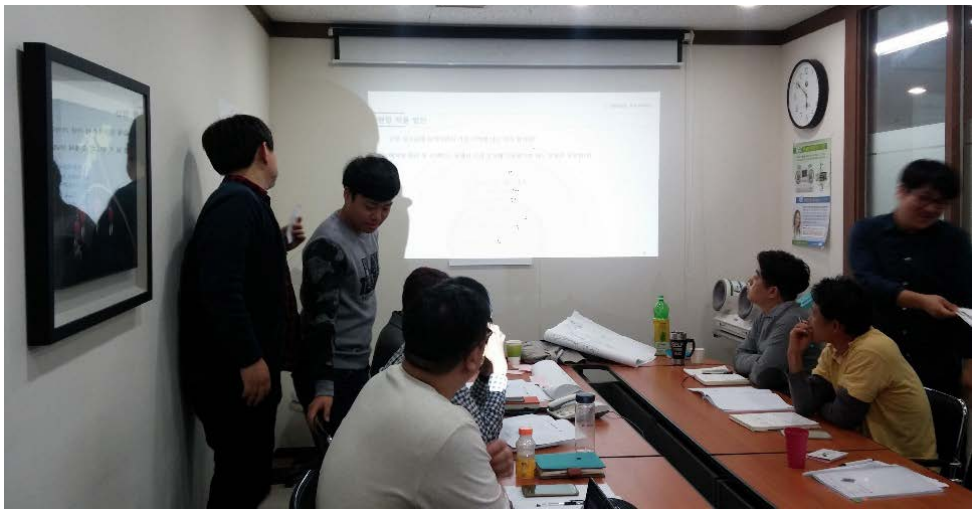
- 경영진과 영업에서 요구하는 서비스 개선 요청이 많이 쌓여있으나 개발팀에서 이를 따라가지 못하고 있음
- 기획서의 완성도가 부족하며 요구변경이 자주 발생함
- 부서간(기획자, 디자인, 개발팀)에 소통이 부족하며 변경 사항에 대한 공유가 잘 이루어지지 않음으로 인해 불필요한 갈등이 발생됨
- 개발자들은 팀장의 지시를 받아서 자기 일만 수행하다 보니 자기 업무 외에는 다른 업무에 대하여 관심이 없음
- 개발자들간에 기술적인 공유가 거의 일어나지 않음
- 개발자들에게 동시에 여러 가지 일을 요청하고 있어 멀티태스킹으로 인한 일정준수가 어려움
- 공통문서 및 코딩주석 등에 대한 표준화 작업이 필요
- 개발팀 업무 진행현황을 개발팀장만 알고 있으며 다른 사람들은 알기 어려움

8.1.4 경량방법론 적용

경량 방법론은 다음과 같은 절차로 프로세스 개선이 진행되었다.

1) 방법론 교육 및 회고

- 기획자를 포함한 전체 개발팀원들을 대상으로 애자일에 대한 개념과 원리, 경량방법론 주요 프랙티스에 대한 구체적인 교육이 진행되었다.

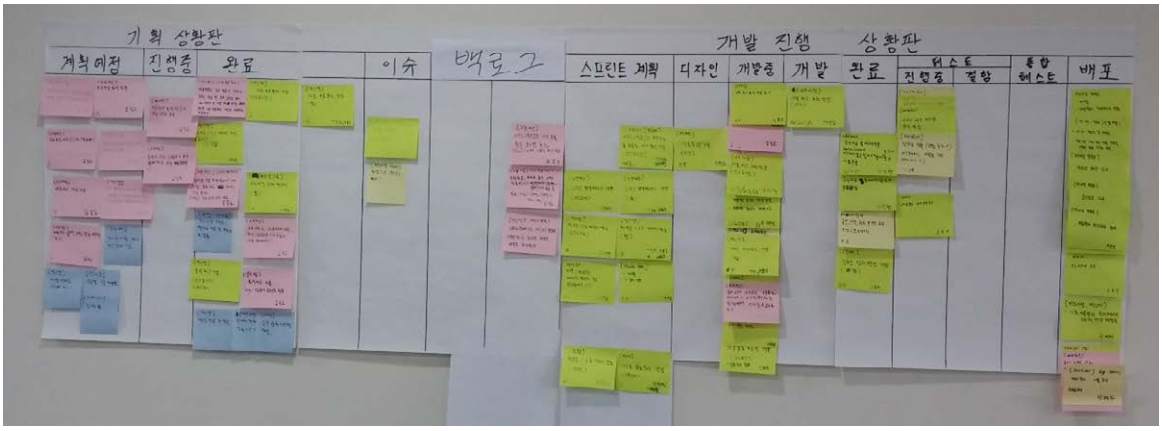


〈 도표 26. 방법론 교육 〉

- 회고를 통하여 기존 업무 프로세스에 대한 문제점과 개선할 사항들이 도출되었다.
- 회고에서 나온 문제점은 앞 절에서 언급되었으며 개선사항은 아래와 같다.
  - 업무 우선순위 미팅 및 제품책임자의 역할 정립
  - 일일 스탠드업 미팅을 통한 파트간 소통 강화
  - 작업 진행 상황판(Task Board) 구축을 통한 업무진행 상황 공유 및 협력
  - 주기적인 이터레이션 계획을 통한 업무 계획
  - 주기적인 회고 수행을 통한 프로세스 개선
  - 서로가 배려하고 존중하는 문화 만들기

2) 업무 흐름에 대한 시각화 및 완료정의

- 전체 업무 진행상황의 가시성을 높이기 위하여 모든 직원들이 볼 수 있게끔 기획팀과 개발팀의 작업 진행 상황판을 벽면에 구축하였다.
- 서비스 기획부터 마지막 배포까지의 업무 흐름도를 작성하고 이를 바탕으로 작업 진행 상황판을 설계하였다.



〈 도표 27. 기획팀과 개발팀 작업 진행 상황판 초기 버전 〉

• 기획팀의 완료정의는 경영진과의 최종 협의가 종료된 것을 기준으로 하였으며 개발팀의 완료정의는 해당 개발자가 사용자 스토리에 대한 테스트케이스가 작성되고 테스팅이 종료된 상태를 기준으로 삼았다.

2) 업무 우선순위 미팅

• 기존에는 영업팀과 경영진, 개발팀장이 모여서 매주 주간업무 미팅을 진행하였는데 2주에 1번은 주간업무 미팅 후에 업무 우선순위 미팅을 추가로 진행하였다. 이 미팅에서는 다음 이터레이션에서 수행할 개발 업무에 대한 우선순위를 결정하였다.

3) 이터레이션 계획

• 이미 기획이 완료된 개발업무들이 쌓여 있는 관계로 2주 단위의 이터레이션 관리를 수행하기로 결정하였다.

• 이터레이션 계획은 요구사항에 대한 이해를 위한 파트1과 개발 태스크 도출과 할당을 위한 파트 2로 나누어 수행되었다.

• 파트1에서는 기획자가 중심이 되어 해당 이터레이션에서 수행할 업무에 대하여 디자이너, 개발자들에게 상세히 설명하고 함께 토론하는 방식으로 진행되었다. 이 미팅을 통해서 개발자들은 요구사항에 대한 이해와 개발 방향을 설정한다.

• 파트2에서는 개발자들이 중심이 되어 스토리 구현을 위한 웹과 안드로이드, IOS, 디자인 등과 같이 기능별 태스크로 분할했다. 태스크로 분할한 이유는 어떤 스토리는 안드로이드 개발만 있는 경우가 있고 어떤 것은 웹, 안드로이드, IOS 개발 모두 포함되어 있는 경우가 있어서 각각의 태스크를 명확히 하자는 목적이 있었다.

• 각각의 태스크에 대해서는 개발자들이 공동으로 평균 작업공수를 추정했다. 예를 들어 안드로이드 관련 작업 태스크는 3명의 개발자가 공동으로 추정하는 방식이다. IOS나 디자인은 각각 1명씩이라 본인들이 추정하고 개발팀장이 검증했다. (평균 작업공수는 잘하는 사람과 못하는 사람의 중간 정도 공수를 의미한다.)

• 기존에 업무 할당은 팀장이 일방적으로 결정하는 방식이었으나 이번에는 가능하면 팀원들이 각자 잘 할 수 있는 업무를 선택하여 자발적으로 가져가는 방식으로 진행했다. 이때 팀장은 팀원들이 본인의 이터레이션내 가용공수만큼 가져갔는지 확인하고 필요시 업무를 균형 있게 분배했다.

4) 일일 스탠드업 미팅

• 기획자와 디자이너, 개발자들은 오전 10:30분에 모여서 20분 정도 스탠딩 미팅을 수행했다. 구성원들은 이 활동을 통해서 서로의 업무를 공유하고 협력하게 되었으며 개인 중심의 일하는 방식에서 팀 중심의 일하는 방식으로 전환하게 되었다. (처음에는 팀원들이 모두 출근하는 10:30분에 진행을 하였으나 업무 집중도를 방해한다고 해서 11:30분으로 늦추어짐)

5) 회고

- 팀 리더의 주관으로 기획자와 디자이너, 개발자 모두 함께 모여서 이터레이션 마지막 날 2시간 정도 회고를 수행했다.
- 회고 방식은 팀 리더가 우선 이터레이션 계획과 실제 수행되었던 결과에 대해서 간략하게 설명하고 잘 진행되지 못했을 경우에는 원인에 대해서 팀원들과 얘기를 나눈다. 이후에는 우리가 잘하고 있는 사항과 개선해야 할 사항, 어떻게 하면 즐겁게 일할 수 있는지에 대한 생각들을 공유했다.

8.1.5 적용 효과

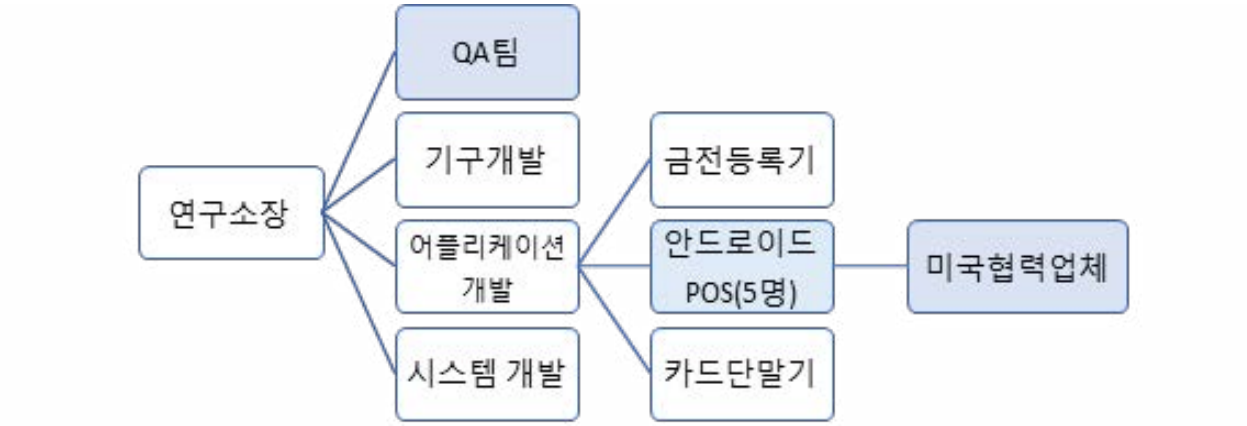
시범 적용 기간이 짧아 아직 효과를 느끼기는 어렵지만 회고시에 나온 내용을 정리하면 다음과 같다.

- 과거에는 팀 리더가 그때 그때 업무를 할당해주다 보니 팀원들이 주도적으로 업무 계획을 세우기가 어려웠다. 하지만 지금은 자기가 해야할 일을 2주 단위로 자발적으로 세울 수가 있어서 좋았다.
- 구성원들 중에 대부분은 일일 스탠드업 미팅을 긍정적으로 생각하게 되었다. 처음에는 이 미팅에 대하여 다소 거부감이 있는 사람들도 있었으나 서로가 업무를 공유하고 필요시 언제라도 도움을 요청할 수가 있어서 다들 좋게 생각했다. 다만 시간이 약간 길어지는 경향이 있어서 그것에 대한 불만이 존재했다. (어떤 때는 30분 가까이 하는 경우도 있었음)
- 업무 진행 상황판은 전체 업무를 가시적으로 볼 수 있고 무엇에 버틀넥이 걸려있는지 바로 알 수가 있어서 좋았다.
- 부서간의 소통이 늘어남으로써 기존에 존재했던 갈등이 많이 줄어들었다.

8.2 S사 사례

8.2.1 조직 개요

이 회사는 POS, 금전등록기, 미니프린터, 카드단말기 등 비즈니스 시장에 필요한 다양한 제품로드맵을 가진 점포자동화 부문의 전문 솔루션업체이다. 내부 조직은 하드웨어, 소프트웨어, 기구를 개발하는 여러팀으로 나뉘어져 있으며 이번에 경량방법론을 적용한 조직은 소프트웨어 개발팀이다. 이 팀은 다시 3개파트(금전등록기, 안드로이드POS, 카드단말기)로 나누어져 있는데 이중에서 안드로이드POS 파트에 적용되었다. 안드로이드POS 파트는시니어 개발자인 파트리더1명과 4명의 개발자가 일하고 있으며 QA팀의 테스터 2명, 미국 협력업체 인력 1명이 함께 협력하고 있다. (개발자들은 7~8년 경력의 2명과 3~5년 경력의 중급개발자2명으로 구성됨)



〈 도표 28. 연구소 내부 조직도 〉

8.2.2 기존 업무 현황 및 프로세스

안드로이드POS 파트는 기존에 있던 POS솔루션을 안드로이드로 전환하는 솔루션을 개발했으며 현재는 이것을 미국에 먼저 적용하며 필드테스트를 진행하는 중이다. 따라서 대부분의 업무는 현재 개발된 안드로이드 POS 솔루션에서 발생되는 결함 해결이 주된 일이다. 결함 수정에는 평균 1~3일정도 소요되는 편이며 주요 업무는 아래와 같다.

- 개발된 솔루션에 대한 결함 해결
- 개발된 솔루션에 대한 기능 추가 및 개선

업무진행은 아래와 같은 절차로 진행된다.

1) 서비스 요청

- 솔루션의 필드테스트를 진행하는 미국 협력업체에서 결함이나 개선사항들이 발생되면 우선순위와 함께 구글닥스에 자세한 내용을 기록한다.
- QA팀에서 테스트 수행시 결함이 발견되면 내부에서 사용하는 이슈관리 도구인 레드마인(Redmine)에 기록한다.

- 현재 많은 양의 개선 요청 사항들이 쌓여 있으며 미국에서 수시로 결함들이 발견되어 빠른 해결이 요구되는 상황임

2) 개발

- 파트리더는 미국에서 보내온 이슈들과 QA팀에서 올린 이슈들을 종합해서 우선순위를 판단하여 팀원들에게 업무를 할당한다.
- 팀원들은 맡은 작업이 완료되면 QA담당자에게 통보하고 트렐로(Trello)를 사용하여 작업 진행 상태를 업데이트 한다.

3) 테스트

- 테스터는 개발자가 완료한 결함이나 기능을 테스트하며 발견된 결함은 레드마인에 기록한다.

8.2.3 주요 문제점

이 곳은 미국 협력업체와 개발팀, QA의 밀접한 협력이 필요한 팀으로 다음과 같은 문제점들이 있었다.

- 우선순위 선정에 대한 일관성이 부족 : 미국 협력업체에서 보내오는 이슈들의 우선순위와 QA팀에서 올리는 이슈들에 대한 우선순위 기준이 다르다보니 개발팀에서 혼란이 발생됨
- 여러가지 도구를 사용하다보니 관리의 비효율성이 발생됨(레드마인, 구글닥스, 트렐로 등)
- 개발팀의 업무분배가 비효율적으로 이루어짐
- 개발팀과 QA팀간의 소통 부족
- 수직적 구조로 인한 구성원간 커뮤니케이션 부족
- 개발일정 예측의 어려움

8.2.4 경량방법론 적용

경량 방법론은 다음과 같은 절차로 프로세스 개선이 진행되었다.

1) 방법론 교육 및 회고

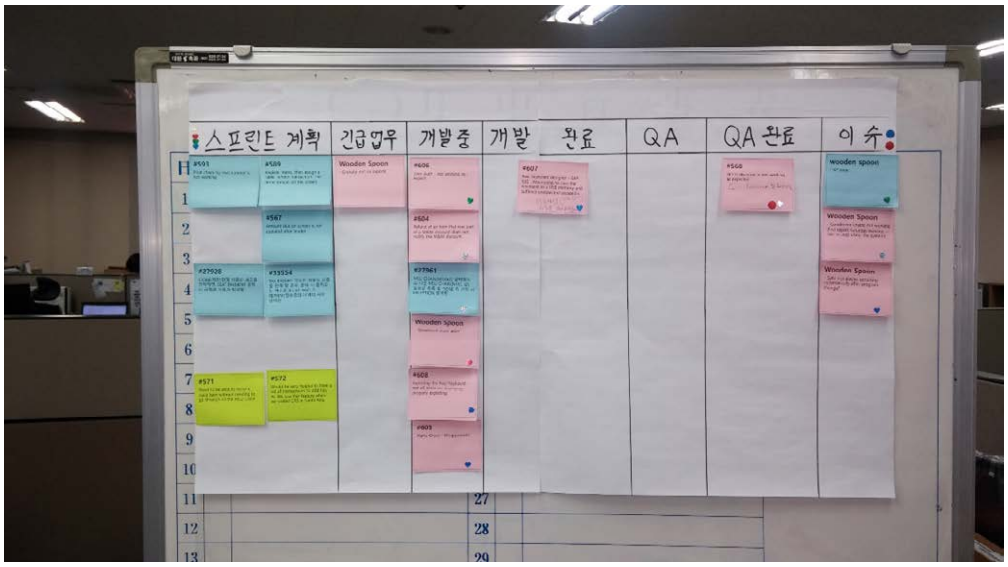
- 어플리케이션 개발팀원들과 QA팀을 대상으로 애자일에 대한 개념과 원리, 경량방법론 주요 프랙티스에 대한 구체적인 교육과 회고를 진행함
- 회고를 통하여 기존 업무 프로세스에 대한 문제점과 개선할 사항들을 도출되었다.
- 회고에서 나온 문제점은 앞절에서 언급되었으며 개선사항은 아래와 같다.

– 개발과정에서 낭비를 최소화 하자(멀티태스킹, 과잉기능 등)
– 파트리더의 소통과 관리 역할 강화(현재는 개발자 역할만 주로 수행)
– 신규기능 개발시 개발과 QA가 함께 참여하여 기획
– 일일 스탠드업 미팅을 통한 파트간 소통 강화

– 작업 진행 상황판(Task Board) 구축을 통한 업무진행 상황 공유 및 협력
– 주기적인 이터레이션 계획을 통한 업무 계획
– 주기적인 회고 수행을 통한 프로세스 개선
– 페어 프로그래밍과 페어테스팅 시도
– 스토리 작성시 완료조건(인수기준) 수립
– 리드타임, 결함발생율 같은 정량화 지표 수립

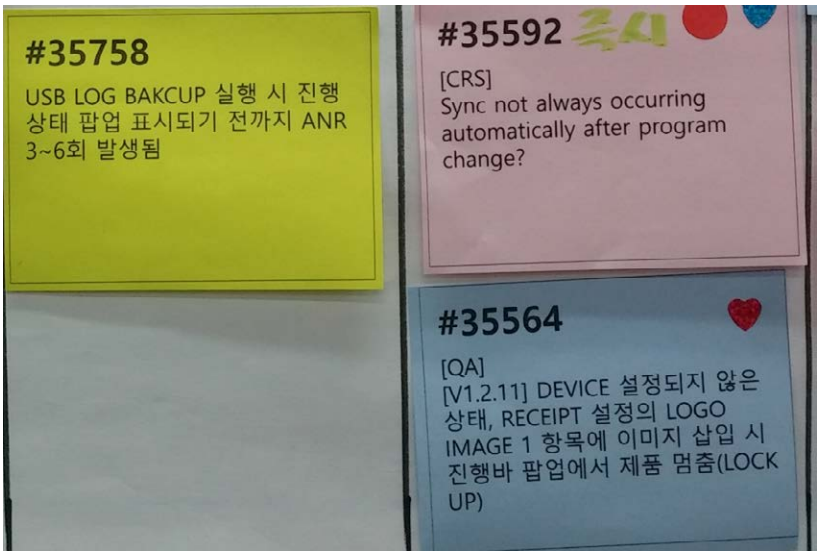
2) 업무 흐름에 대한 시각화 및 완료정의

• 전체 업무 진행상황의 가시성을 높이기 위하여 모든 직원들이 볼 수 있게끔 개발팀과 QA팀의 작업 진행 상황판을 벽면에 구축하였다.



〈도표 29. 작업 진행 상황판 초기 버전〉

• 상황판의 두번째 '긴급업무' 는 미국 협력업체에서 수시로 요청하는 이슈들을 처리하기 위하여 만들어짐 (야간에 많은 이슈들이 등록되는 상황임)
• 개발완료 정의는 개발자가 해당 스토리에 대한 수정을 완료하고 테스트 케이스(사전조건, 동작순서, 기대결과 작성) 작성하는 것으로 정의함
• QA완료는 해당 기능에 대한 회귀테스트까지 수행한 것으로 정의함



〈 도표 30. 스토리 작성 사례 〉

3) 이터레이션 계획

- 현재 작업해야 할 많은 이슈들이 쌓여 있는 상태라 주간 단위로 이터레이션 계획을 수립하기로 결정함
- 매주 금요일 오후에 다음주에 수행할 작업항목들을 팀 리더가 중심이 되어 팀 캐퍼의 20%정도만 선정함 (나머지 80%는 매일 들어오는 이슈들에 대한 해결을 수행해야함)
- 선정된 항목에 대한 스토리포인트를 추정

4) 일일 스탠드업 미팅

- 매일 9시반에 15~20분간 개발팀과 QA테스터가 참여하여 업무 진행상의 이슈를 공유하고 협력함
- 밤사이에 미국에서 들어온 이슈들은 개발리더가 정리하여 우선순위를 설정하고 작업상황판의 '긴급업무'에 게제함
- 과거에는 업무할당을 파트 리더가 직접 하였으나 이번에는 팀원들이 우선순위에 따라 자발적으로 가져가는 것을 원칙으로 하고 필요시 리더가 조정을 수행함

5) 회고

- 파트 리더의 주관으로 개발자 및 QA 담당자들이 모두 함께 모여서 매주 금요일 오후에 1시간 정도 수행했다.
- 회고 방식은 팀 리더가 우선 이터레이션 계획과 실제 수행되었던 결과에 대해서 간략하게 설명하고 잘 진행되지 못했을 경우에는 원인에 대해서 팀원들과 얘기를 나눈다. 이후에는 우리가 잘하고 있는 사항과 아쉬운 점, 개선해야 할 사항 대한 생각들을 공유했다.

6) 기타

- 여러가지 도구를 사용하다보니 관리의 비효율성이 발생되어 레드마인(Redmine)으로 통일하기로 결정함 (트렐로는 작업상황판으로 대체)
- 미국 협력업체에게 이슈를 기록하는 자세한 방법과 함께 구글닥스 대신 레드마인을 사용할 수 있도록 요청함
- 정적분석 도구인 Findbug 도입을 시작하였으며 지속적인 통합은 향후 적용하기로 결정함

8.2.5 적용 효과

시범 적용 기간이 짧아(1달 정도) 아직 효과를 느끼기는 어렵지만 회고시에 나온 내용을 정리하면 다음과 같다.

- 작업 진행상황판을 통하여 팀간, 담당자간의 업무 진행 현황을 한눈에 알아 볼 수 있는 것이 좋았다.
- 업무의 유형이 구분되어 있어 우선적으로 해야 할 업무의 판단이 쉬워짐
- 매일 진행되는 스탠드 미팅을 통하여 담당 업무의 의견 전달이 명확해짐
- 과거보다 QA와 협력이 잘되어 이슈를 빨리 해결하는 것 같다.
- 일일 미팅을 통하여 자잘한 회의들이 많이 없어졌다.

8.3 적용 교훈

- W사 같은 경우 평상시에 팀원들은 기존 솔루션에 대한 문의 및 장애 응대, 결함 해결 등을 병행하고 있었기에 업무 할당은 가용공수의 70% 정도만 할당하고 30%는 여유시간을 주었다. 예를들어 OOO팀원의 이터레이션 내 가용공수가 10일이라면 하루는 각종 미팅으로 인하여 워킹데이에서 빼고 9일간의 워킹데이 중 7일 정도만을 이터레이션 계획시 할당했다.

- W사 같은 경우 사용자 스토리에 대한 포인트 추정은 처음에 시도해보았으나 팀원들이 어려워하여 일단 평균 투입공수만 추정하고 향후에 추정을 시도하기로 결정했다.

- S사는 미국 협력업체에서 발생되는 이슈 해결이 주된 업무이다보니 매일 아침 업무의 우선순위를 결정하고 이를 빠르게 해결하는 쪽에 적용의 초점을 맞추었다.

- 경량 방법론에는 다양한 프랙티스들이 있으나 이런 활동들을 한꺼번에 적용하는 것은 바람직하지 않은 것 같다. 현재 시점에서 해당 조직에 가치가 있는 것을 중심으로 점진적으로 도입하는 것이 바람직하다.

- 오래된 습관처럼 팀원들이 개인 플레이에서 팀 플레이로 바로 가기는 어려운 것 같다. 이것을 변화시키기 위해서는 팀 리더가 팀원들에게 팀플레이를 하도록 지속적으로 독려하는 활동이 필요해보인다.

- 회고 및 이터레이션 계획 등을 주기적으로 수행하기 위해서는 주당 0.5일은 이런 회의를 위한 시간으로 비워두는 것이 좋다. 만약 2주의 이터레이션을 수행한다면 최소 1일~1.5일은 이런 미팅 시간으로 할애한다. (워킹데이에서 빠져야 한다는 의미) 이런 미팅이 끝나고 남는 시간은 팀원들이 하고 싶은 일을 할 수 있도록 자율성을 주는 것이 바람직하다.

APPENDIX 01

용어해설

APPENDIX 01 용어해설

용어	정의
인수 테스트	제품 책임자 및 사용자에 의해서 수행되는 테스트. 사용자 스토리의 "릴리즈 여부"를 결정하며, 소프트웨어가 해당 요구 사항을 충족하는지를 수락하는 테스트
인수 테스트 주도 개발	인수 테스트에 통과하기 위해 코드를 개발하는 접근법. 요구되는 행위 및 테스트를 위해 다음과 같은 특정한 포맷과 특정한 테스트 자동화를 요구할 수 있음 Given 어떤 상황이 주어지면 When어떤 이벤트가 발생할 때 Then 이것이 발생함
자동화된 빌드& 테스트	자동화된 빌드와 테스트는 지속적인 통합을 지원하는 데 사용된다. 코드가 빌드되고 테스트하는 것을 자동화하는 개발 접근법
자동화된 단위/ 회귀 테스트	코드가 제공되고 빌드될 때마다, 회귀 및 통합 문제가 없음을 보장하기 위해, 단위 테스트와 단위 회귀 테스트가 자동으로 수행된다. 새로 제공된 코드가 아무 문제 없이 기존 코드와 함께 잘 동작하는 것을 확인
행동 주도 개발	기능이 구현되기 전에 인수 테스트에 기반한 개발 접근법. FitNesse 와 Cucumber 과 같은 도구를 사용해 자동화된다. 애자일 테스팅에 좋은 방법이다. 인수 테스트 주도 개발 (ATDD)의 다음 성숙 단계
체크리스트	체크리스트는 다양한 경험을 통해 품질을 향상시킨다고 입증된 기술이다. 효과적인 체크리스트는 정확성 보다는 이슈 영역을 고려할 수 있는 부울(Boolean) 질문의 짧은 목록을 구성한다. (부울 질문은 참 또는 거짓으로만 응답할 수 있다.) 체크리스트는 문제점을 찾지 못하는 질문은 제거를 하면서 시간이 지남에 따라 업데이트를 해야 함
코딩 우선 프로그래밍	프로그래밍과 단위 테스트에 대한 코딩 우선 접근법은 코드를 먼저 개발하고 코드가 잘 동작하는지를 확인하기 위해 단위 테스트를 수행
지속적 전달	빌드와 단위 테스트 및 단위 회귀 테스트가 성공할 때마다 전체 회귀 테스트를 위해 빌드 및 새 코드가 테스트 서버에 배포된다. 모든 것이 정상이면 자동화 된 배포(아직 수동 개입이 있음)가 시작된다. 배포를 위해 새 코드의 유효성을 검사하기에 적절한 회귀 테스트 세트가 필요하다. 지속적 통합의 다음 성숙 단계
지속적 배포	지속적 전달과 자동화된 배포(수동 개입 없음). 자동화된 체크는 문제를 식별하고, 배포를 멈추고, 수동 개입을 트리거하는 데 사용된다. 시장 진입 시간을 단축하고 롤백이 가능하다. 지속적 전달의 다음 성숙 단계
지속적인 통합	새 코드가 공유된 저장소에 제공될 때마다 자동화된 빌드는 기존 코드와 통합될 수 있는지를 확인
지속적 프로세스 개선	프로세스 개선은 보통 회고를 통해, 각 이터레이션이나 릴리즈 말에 수행된다. 하지만 서로 다른 빈도일 수 있다. 예를 들어 4주 마다 할 수 있음
COTS	상용 제품. 사용자가 최소한의 IT공급자의 지원을 받으며 설치 및 실행할 수 있는 IT기관 밖에서 고객에게 전달되는 소프트웨어

결함 보고	소프트웨어를 테스트하거나 사용하는 누군가가 인수 기준 또는 완료 정의를 충족하지 못하는 문제를 발견할 때, 그 문제는 팀에게 보고될 필요가 있고 해결책이 합의되고 수행될 필요가 있음
정해진 이터레이션 기간	이터레이션은 4주 이하임. 이터레이션이 짧아지면, 릴리즈가 빨라지고 피드백도 빨리 받을 수 있음. 하지만, 이터레이션과 연관된 간접비(이터레이션 계획, 회고)가 긴 이터레이션에 비해 더 높을 수 있음
완료 정의	릴리즈 가능한 기준 정의. 인수 기준을 충족시키는 것을 포함하지만 엔지니어링 전제조건이 완료되었음도 포함 (예를 들어 코드 리뷰, 단위 테스트 커버리지, 정적 분석)
준비 정의	사용자 스토리를 구현하기 전 반드시 충족돼야 하는 기준.
에픽	사용자 스토리보다 큰 여러 관련 기능을 설명하는 규모가 큰 요구사항. 에픽은 일반적으로 단일 이터레이션으로 구현하기에는 너무 크기 때문에 스토리로 분할해야 한다. 사용자가 높은 수준에서 원하는 것을 설명 할 때 유용하다. 그러나 개발자가 에픽을 여러 이터레이션으로 쪼개는 것은 쉽지 않음
탐색적 테스트	학습과 동시에 테스트 설계, 수행을 함. 테스트 결과에 따라 다음 테스트 영역이 탐색된다. 탐색적 테스트 동안에 테스트 과정, 결과, 결론, 잠재적인 결함 등을 충분히 기록
빠른 사용자 피드백	반복 점진적 접근법의 이점. 짧은 작업 반복 동안에 작동하는 것과 작동하지 않는 것에 대해 사용자는 빠르게 의견을 줄 수 있음
기능 주도 개발	팀은 아키텍처 구조보다 기능을 구현함. 기능이 이미 충분히 통합되었으므로, 통합 단계의 필요성 제거됨
점진적인 개발	전달할 때마다, 사용자를 위한 더 많은 기능이 전달됨. 고객 및 사용자는 한번에 모든 것이 아니라 점진적으로 전달하는 것을 동의해야 함
비공식적 결함 관리	현 이터레이션에서 이슈가 수정될 수 있을 때는 이를 문서화할 필요가 없음. 단기적인 이슈임으로 수정 사항에 대한 기록이 없음. 이 기록은 사용되지 않음으로 문제가 안됨
정보-프로젝트 계획서	시각적이고 종종 도표로 돼 있으며, 프로젝트의 핵심 성공요인 관점이 사무실 벽에 진열됨. 도표, 마인드맵 등이 사용될 수 있음. 제품 책임자와 팀원이 프로젝트에 익숙하다면 필요하지 않을 수 있음
정보-릴리즈 번-업 차트	번-업 차트는 이터레이션의 진행상황에 대한 명확한 관점을 팀 또는 다른 사람에게 제공함. X축은 시간의 경과를 보여주며 Y축은 남아있는 작업을 보여줌. 는 번-업 차트를 권장함
정보-작업 현황판	작업 현황판은 프로젝트의 현 상태에 대한 명확한 관점을 팀 또는 다른 사람들에게 제공함. 실질적인 작업 현황판이 가장 효과가 있으며, 만약 팀 밖의 사람이 정보를 필요로 하거나, 팀을 방문할 수 없다면 전자 버전과 함께 사용됨
이슈	이슈는 잘못 진행된 것이다- 리스크요인이었는데 발생하였고 그 결과를 처리해야 함. 이슈는 조직, 프로젝트, 팀, 제품 결함 등이 있음
이터레이션, 타임박스	이터레이션은 확정된 기간임. 확정된 기간은 개발이 일정하게 진행됨을 의미. 시간제한이 돼 있지 않으면, 이터레이션 길이가 각 이터레이션에서 합의돼야 함. 시간제한이 있는 이터레이션을 이터레이션이라고 불림
반복적 개발	처음부터 제대로 하기를 기대하진 않지만, 프로젝트가 진행됨에 피드백과 변화하는 요구사항에 대응함. 이는 프로젝트가 여러 번 반복됨에 따라 요구사항이 발전되게 함을 의미. 비 반복적 개발은 모든 요구사항을 미리 알고 있는 경우에만 유용할 수 있는데 이는 매우 드묾
회의 - 백로그 유지	백로그 항목이 최신 상태로 유지되고 완료되었는지를 확인하는 회의. 제품 책임자에게 남겨질 수 있음으로 정기적으로 수행하지 않거나 개발자의 관련 기술 정보를 포함할 수 없음

회의 – 일일 스탠드 업	15분간의 미팅으로 팀원 모두가 무슨 일이 일어난 지 알 수 있음. 종종 세 가지 질문(한 것 /해야 할 것/장애물) 방식을 사용함. 너무 길어질 수 있으며, 참여자들이 멍해질 수 있고, 다른 사람이 이야기하는 내용을 무시할 수 있으므로 적절히 중재할 필요가 있음
회의 – 이터레이션 계획	제품 책임자와 팀 간에 이터레이션에 구현될 내용에 동의하는 회의(일반적으로 반나절). 팀이 자신의 속도(이터레이션 당 구현할 수 있는 항목 수)를 알아야 함. 제품 책임자와 팀간에 이터레이션 동안 달성해야 하는 것에 대한 합의를 얻는 것이 중요함
통합되지 않은 테스트	용어 해설 참조: 테스트 접근법들, 대안
가끔 수행하는 테스트 이터레이션	용어 해설 참조: 테스트 접근법들, 대안
가끔 수행하는 테스트 이터레이션	용어 해설 참조: 테스트 접근법들, 대안
짝 프로그래밍	두 명의 프로그래머가 하나의 컴퓨터를 공유함. 새로운 팀원을 참여시키기에 좋으며, 감소된 수정과 재 테스트가 고려될 때 더 효율적임. 하지만 이것이 낭비라고 느끼는 경영진은 좋아하지 않음
짝 테스트	두 명의 테스터가 하나의 컴퓨터를 공유함. 새로운 팀원을 참여시키기에 좋으며, 탐색적 테스트에서 테스트 아이디어를 내고, 이슈를 조사함
병렬 테스트 이터레이션	용어 해설 참조: 테스트 접근법들, 대안
페르소나	시스템의 주요 사용자를 정의하기 위해 사용자 예시를 이용– 개발과 테스트를 위해 이용될 수 있음. 사용자 인터페이스 설계 및 테스트에는 매우 유용하지만 애자일 방법론의 필수적인 사항은 아님
계획된 이터레이션 내용	이터레이션의 내용이 합의되면, 추가적인 항목이 더해지거나 제거될 수 없음. 제품 책임자, 사용자 또는 고객이 지속적으로 항목을 추가하고 팀에 과부하가 걸리지 않도록 하지만, 팀이 변경에 덜 반응적일 수 있음. 계획된 내용이 변경되면 스크럼 마스터나 팀 리더를 통해서 합의돼야 함
우선순위가 지정된 제품 백로그	제품 백로그는 우선순위가 정해지므로, 항목을 우선순위에 따라 작업함. 타임 박스 방식이 잘 작동하면, 이터레이션에 계획된 모든 항목이 전달되므로 이터레이션 내의 우선순위 지정이 중요하지 않음. 제품 책임자가 이터레이션 내용에 동의하면, 계획 활동을 돕기 위해 우선 순위 지정만 필요함. 제품 백로그는 제품에서 필요할 수 모든 항목의 정렬된 목록이며 제품에 대한 변경 사항에 대한 단일 요구 사항임. 제품 책임자는 내용, 가용성, 순서를 포함해 제품 백로그를 담당함
제품 백로그 관리	제품 책임자는 백로그에서 항목을 추가 및 제거하고, 릴리즈 그룹으로 백로그를 우선순위로 지정하며, 고객 및 사용자와 함께 프로젝트 방향의 변경을 논의하면서 백로그 관리를 함. 제품 책임자는 가장 중요한 요구사항을 위해 사용자 스토리를 작성함. 그 요구사항에는 사용자 요구, 팀의 기술적 요구, 다음 이터레이션에서 수정할 결함을 포함함. 제품 백로그 관리에는 선택적으로 고객 및 사용자와의 정기 회의가 포함돼 우선순위를 검토하고 이터레이션에서 잘 된 점, 개선 된 점 및 프로젝트 수행 개선을 위한 제안을 리뷰함
제품 책임자	고객을 대표하는 요구 사항, 인수 등에 대한 모든 결정을 담당하는 단일 연락 창구이자 권한이 제품 책임자의 역할임. 팀 구성원이 정보 및 피드백을 빨리 얻을 수 있음. 이 역할을 수행하는 데 필요한 시간과 중요성을 이해할 수 있도록 비즈니스에 대한 교육이 필요함
제품 책임자– 지속적 상호작용	제품 책임자는 팀의 일원이 아니지만, 고객과 사용자의 대표로서 매일 팀에 참여해 지속적인 고객 의 요청을 제공함

제품 책임자–제품 백로그	제품 책임자는 제품 백로그에 대한 책임을 진다. 제품 백로그는 이터레이션 백로그에 아직 없는 요구되는 모든 것의 항목임. 다음 이터레이션 백로그를 위한 항목은 제품 백로그로부터 선택됨
리팩터링	기능은 그대로 유지하면서 시스템 아키텍처 및 코드의 유지 보수성, 성행 등을 개선하는 활동
회귀 테스트	이터레이션에서의 변경이 이전에 수행된 작업에 나쁜 영향을 미치지 않음을 보장하기 위한 테스트. 변경은 소프트웨어나 인프라에 대한 변경임. 회귀 테스트 세트는 매우 빠르게 커지며 사용자 스토리 회귀 테스트의 작업량을 관리하는 여러 방법이 있음. 리스크가 높은 사용자 스토리 테스트를 자동화하고 모든 이터레이션에서 수행함 자동화된 단위 회귀 테스트를 포함시킴 이터레이션 내의 리스크에 따라 수행할 회귀 테스트 상세 항목을 작성함
모든 이터레이션에서 릴리즈 가능한 기능	고객이 기능을 릴리즈하지 않아야 한다고 결정할 수도 있지만, 각 이터레이션이 끝날 때 릴리즈 가능한 기능이 있어야 함. 그래야 진행률을 측정할 수 있음. 미완성된 기능은 백로그로 돌아가며, 다음 이터레이션에서 해결되는지에 대한 새로운 결정이 이루어짐
릴리즈 계획	각 이터레이션에서 사용 가능한 코드를 제공하는 '이상적인' 애자일 접근법이 사용된다면 일반적으로 이터레이션 계획을 보완하기 위해 별도의 릴리즈 계획 활동이 거의 필요 없음. 그러나 많은 프로젝트에서, 릴리스가 모든 이터레이션에서 수행되지는 않으며, 이러한 경우 별개의 릴리즈 계획이 요구됨
회고	이터레이션이 끝나면 무엇이 잘 진행되었는지, 무엇이 더 잘 진행돼야 하는지를 파악하기 위해 열리는 회의
리스크	리스크는 일어날 가능성이 있는 아직 발생하지 않은 잠재적인 이슈이며, 만일 발생하면 악 영향을 미침. 용어 해설 참조: 이슈
간결한 설계	미래에 유용 할 수 있는 것이 아닌, 지금 필요한 것을 기반으로 설계로 실용적이어야 함
이터레이션 백로그	이터레이션에서 완료돼야 하는 목록. 사용자 스토리와 인수 기준으로 구성됨
이터레이션 계획	이터레이션 계획은 이터레이션 초에 열리는 회의로 팀과 제품 책임자에 의해 수행됨. 어떤 스토리가 이터레이션 백로그에 포함될지, 어떻게 그것을 수행할지를 결정함
이터레이션 회고	용어 해설 참조: 회고
이터레이션 제로	애자일 프로젝트가 시작할 때 모든 것이 준비되고 릴리즈 가능한 기능을 개발할 수 있도록 프로젝트를 수립하는 초기 활동. 일반적으로 사무실 공간 설정, 개발, 테스트 프로세스, 팀원 선발, 환경과 도구 설치와 같은 활동을 포함함. 이터레이션 제로가 필요한가와 무엇이 포함돼야 하는지에 대해 논쟁이 있다. 예를 들어, 아키텍처의 초기 정의를 포함해야 한다고 주장하는 사람들이 있음
이터레이션의 길이, 권장사항	이터레이션은 4주 보다 길어질 수 없음. 더 짧은 기간의 이터레이션으로 릴리즈가 되면, 더 빨리 피드백을 받을 수 있지만, 이터레이션 관련 간접비가(예를 들어, 이터레이션 계획, 회고) 더 긴 이터레이션보다 높을 수 있음
스토리 포인트	스토리의 상대적인 크기를 측정하는 방법으로, 이를 비교하고 노력, 속도 및 진행을 추정하는 데 사용할 수 있음. 스토리는 상대적 측정 일 뿐이므로 공수와는 직접적으로 관련이 없음
지속 가능한 속도	팀은 무기한으로 지속될 수 있는 속도로 작업하므로, 최적의 작업을 할 수 있도록, 개발자와 테스터가 초과 근무나 장시간 근무를 피하기 위해 작업량을 관리해야 함
테스트 주도 개발	코드가 작성되기 전에 단위 테스트 테스트 세트를 먼저 작성하는 개발 접근법. 많은 프로그래머들은 이 접근법으로 수행하는 것이 어렵다고 생각하지만, 이 방법은 코드가 단위 테스트에 의해 보장되며, 단위 회귀 테스트도 보장됨

팀 룸	팀 내 의사소통을 장려하고 팀 밖의 소음으로부터 보호해주는 사무실 공간이 필요함
팀 – 공동 책임	팀원은 코드를 변경할 수 있음. 만일 팀원이 변경 요구가 필요하면, 개발 과정을 늦추기보다는 책임을 지고 변경시키도록 허용 돼야 함
팀 – 함께 있음	팀원 전부는 반드시 한 방에 함께 위치해야 함. 이는 의사소통과 팀 작업에 필요함
팀 – 교차기능적	릴리즈 가능한 소프트웨어를 구현하는 데 필요한 모든 기술이 팀 내에 있어야 함. 팀은 개발자, 테스터, 비즈니스 분석가(필요 시), 운영자, 팀 리더로 구성됨. 전문가들도 필요할 수 있음
팀 – 크기 – 7±3명	한 팀은 너무 많은 의사소통이 존재하지 않도록 팀 리더를 포함해10명을 초과하면 안 됨
팀 – 전문가	성능 및 보안 테스트와 같은 전문 기술은 항상 필요하지 않음. 팀은 필요 시 전문가 도움을 받을 수 있음
팀 – 팀 리더	매일 팀을 관리하는 가장 숙련되고 자격이 충분한 사람. 애자일 주요 개념은 팀이 반드시 자기조직화해야 한다는 것인데 모든 팀원과 경영진의 참여가 있어야 함
기술 부채	기술적 부채는 애자일 팀 결과물의 품질 부족에 대한 비유임. 품질 부족은 개발 및 테스트 중에 의도적인 타협으로써 발생함. 예를 들어 개발자와 테스터가 서둘러서 또는 필수적 기술이 부족해 실수하거나 차선의 결정을 내릴 때 발생할 수 있음
테스트 접근법, 대안: 통합되지 않은 테스트– 하이브리드	권장하지는 않지만 필요할 수 있음. 독립적인 병렬 테스트 팀에 사용되며, 보안 테스트와 같은 전문 테스트가 수행될 때 테스트 이터레이션에서 사용됨
테스트 접근법, 대안: 가끔 수행하는 전문가 테스트 이터레이션	전문가 테스트는 개발 이터레이션을 따르는 특별한 테스트 이터레이션에서 가끔 수행됨. 일반적으로 각 이터레이션에 쉽게 제공될 수 없는 테스트 환경을 필요로 하는 것과 전문가 테스트 활동이 수행됨. 완료 정의는 개발 이터레이션에만 적용되지는 않음.
테스트 접근법, 대안: 가끔 수행하는 테스트 이터레이션	권장하지는 않지만 필요할 수 있음. 병렬 테스트 이터레이션 수행에 대한 대안은 몇 개의 개발 이터레이션 후에 이터레이션 팀이 이전 개발 이터레이션에서 전달 가능한 것에 대해 '테스트 이터레이션'를 수행하는 것임
테스트 접근법, 대안: 병렬 테스트 이터레이션	권장하지는 않지만 필요할 수 있음. 병렬 테스트 이터레이션은 별개의 팀으로서 작업하면서 개발 팀보다 약간 뒤처져 있음. 테스트 팀은 먼저 준비해 개발 이터레이션의 결과물을 테스트 하고, 개발 팀으로 결함 보고서를 보냄. 따라서 개발자는 이터레이션에서 수행 할 두 가지 뚜렷한 작업을 수행함. 개발과 결함 수정을 해야 함. 결국 두 버전을 동시에 사용해야 하므로 형상관리와 개발 및 테스트 환경 관리가 더욱 어려워짐. 디버깅 된 코드가 다시 테스트되며 최종적으로 시스템은 인수 및 회귀 테스트를 거쳐 사용자에게 제공됨
테스트 접근법: 통합된 테스트	권장됨: 이상적인 이터레이션 내에서 모든 테스트는 함께 위치한 팀에 의해 그 이터레이션 이내에 수행됨. 개발자와 테스터가 서로를 동등하게 대하면서 결과물의 품질에 공동 책임을 지며 작업하는 통합된 접근법. 통합된 접근법은 대안 접근법들보다 훨씬 더 생산적임
테스트 접근법: 병렬 테스트 이터레이션	테스트는 별개의 팀에 의해 개발 이터레이션을 바로 뒤따르는 테스트 이터레이션에서 수행됨. 테스트 독립성을 유지할 수 있으며, 만일 테스트를 외주 할 경우, 유용할 수 있지만, 교차기능성, 의사소통, 타임투 마켓 장점 놓칠 수 있음. 완료 정의는 개발 이터레이션에만 적용되지는 않음. 때때로 미성숙한 애자일 조직에서 완전한 교차기능 팀으로 가는 과정에 사용 될 수 있음
테스트 차터	테스트 차터는 테스트의 목표, 리스크, 접근법, 시간 계획, 수행되는 실제 테스트를 작성하는 양식임. 탐색적 테스트에 사용됨

테스트 세트	요구되는 테스트 준비, 테스트 단계, 예상 결과를 기술하는 테스트케이스의 묶음. 일반적으로 테스트 세트를 작성한 테스터가 아닌 다른 누군가가 테스트를 수행할 수 있도록 만들어짐. 사용자가 인수 테스트 할 때 사용하기 위해 작성된다.
타임 박스	활동에 일정한 시간을 허용하는 방법. 타임 박스된 기간이 완료되면 활동은 중지돼야 함. 타임박스는 팀이 시간에 맞게 작업을 계획하고 수행하도록 적절한 수준을 유지하도록 권장함
이터레이션	이터레이션은 변경되지 않는 반복 기간임. 설정된 길이의 반복을 통해 개발자와 고객은 개발을 일정하게 진행할 수 있음. 타임 박스가 없다면 반복의 길이가 반복 될 때마다 합의되어야 함
사용자 스토리	요구되는 기능을 정의하는 방법. 최소한의 상세 정보가 포함돼 있으므로 지속적인 고객 참여를 통해 지원이 필요하지만 작성 및 사용법은 쉬움. 사용자 스토리는 프로젝트에서 구현할 가치가 있는 기능을 설명함. 주로 사용자나 제품 책임자로부터 나오지만 시스템을 구현하는 데 필요한 사항(예: 테스트 도구, 인프라 변경, 리팩토링, 결함 수정, 기술 부채 해결)으로부터 나오기도 함
사용자 스토리 – 인수 기준	인수 기준은 사용자 스토리 완료의 주요 척도를 제공함
사용자 스토리 – 역할-기능-이유	사용자 스토리를 기술하기 위한 템플릿: · xxx로서/ · 나는 yyyy를 원한다. /zzzz 하기 위해. 사용자 스토리가 가치 있는 무언가를 사용자에게 제공한다는 것을 보장하는 데 좋으며, 세부적인 사용자 스토리를 요구하는 사람에게 좋음
사용자 스토리 – 스토리 카드	사용자 스토리에 대한 정보를 담고 있는A6 카드. 요구되는 사용자 스토리 정보를 정의된 포멧으로 사용할 수 있음
사용자 스토리 – 업무	사용자 스토리를 한 사람이 하루 만에 수행 할 수 있는 작은 부분으로 쪼개는 방법. 의사 소통이 잘되지 않는 서로 다른 팀원에 의해 작성되는 경우에 유용함
UX	사용자 경험. UX는 최종 사용자가 회사, 서비스 및 제품과 상호 작용하는 모든 측면을 다루며 소프트웨어 사용시 사용자가 의미 있는 경험을 할 수 있도록 지원함. 사람과 기술의 상호 작용은 기계에 의한 것이라기보다는 사람에 의한 것임. UX 방법을 사용하면 제품의 기능적 측면과 기술적 측면뿐만 아니라 신뢰성, 접근 가능성, 안전성, 신뢰성 및 제품에 대한 정서적 반응에도 초점을 맞춤으로써 소프트웨어를 보다 효과적으로 개발할 수 있음. 이는 소프트웨어를 의미 있고 즐겁게 사용하게 하고 효율성과 효율성을 측정하는 척도임
속도	일반적으로 이터레이션 당 완료되는 스토리 포인트 관점의 속도. 이터레이션 계획 및 릴리즈 계획을 지원하는 데 필요함

APPENDIX 02

힌트와 팁

APPENDIX 02 힌트와 팁

APPENDIX 03

예제목록

APPENDIX 03

예제목록