Certified Tester

Foundation Level Extension Syllabus

Agile Tester

(ISTQB 애자일 테스터 한글 실라버스)

ISTQB CTFL_AT_v.2014_Kr1.0_201505_KSTQB

International Software Testing Qualifications Board

Foundation Level Extension Agile Tester Working Group: Rex Black (Chair), Bertrand Cornanguer (Vice Chair), Gerry Coleman (Learning Objectives Lead), Debra Friedenberg (Exam Lead), Alon Linetzki (Business Outcomes and Marketing Lead), Tauhida Parveen (Editor), and Leo van der Aalst (Development Lead).

Authors: Rex Black, Anders Claesson, Gerry Coleman, Bertrand Cornanguer, Istvan Forgacs, Alon Linetzki, Tilo Linz, Leo van der Aalst, Marie Walsh, and Stephan Weber.

Internal Reviewers: Mette Bruhn-Pedersen, Christopher Clements, Alessandro Collino, Debra Friedenberg, Kari Kakkonen, Beata Karpinska, Sammy Kolluru, Jennifer Leger, Thomas Mueller, Tuula Pääkkönen, Meile Posthuma, Gabor Puhalla, Lloyd Roden, Marko Rytkönen, Monika Stoecklein-Olsen, Robert Treffny, Chris Van Bael, and Erik van Veenendaal; 2013-2014.

본 한글 실라버스 제작에 소중한 시간을 할애하여 번역, 리뷰, 검수, 편집 작업을 진행하고 도움을 주신 윤진우님, 송홍진님, 김모세님, 박계홍님, 김희선님, 박준표님, 황상철님, 조현길님, 강성훈님께 감사 드립니다.

- Korean Software Testing Qualifications Board

Table of Contents

0. Introduction to this Syllabus

0.1 Purpose of this Document

This syllabus forms the basis for the International Software Testing Qualification at the Foundation Level for the Agile Tester. The ISTQB® provides this syllabus as follows:

- To National Boards, to translate into their local language and to accredit training providers. National Boards may adapt the syllabus to their particular language needs and modify the references to adapt to their local publications.
- To Exam Boards, to derive examination questions in their local language adapted to the learning objectives for each syllabus.
- To training providers, to produce courseware and determine appropriate teaching methods.
- To certification candidates, to prepare for the exam (as part of a training course or independently).
- To the international software and systems engineering community, to advance the profession of software and systems testing, and as a basis for books and articles.

The ISTQB® may allow other entities to use this syllabus for other purposes, provided they seek and obtain prior written permission.

0.2 Overview

The Foundation Level Agile Tester Overview document [ISTQB_FA_OVIEW] includes the following information:

- Business Outcomes for the syllabus
- Summary for the syllabus
- Relationships among the syllabi
- Description of cognitive levels (K-levels)
- Appendices

0.3 Examinable Learning Objectives

The Learning Objectives support the Business Outcomes and are used to create the examination for achieving the Certified Tester Foundation Level—Agile Tester Certification. In general, all parts of this syllabus are examinable at a K1 level. That is, the candidate will recognize, remember, and recall a term or concept. The specific learning objectives at K1, K2, and K3 levels are shown at the beginning of the pertinent chapter.

1. 애자일 소프트웨어 개발 - 150 분

키워드

애자일 선언, 애자일 소프트웨어 개발, 점진적 개발 모델, 반복적 개발 모델, 소프트웨어 수명주기, 테스트 자동화, 테스트 베이시스, 테스트 주도 개발, 테스트 오라클, 사용자 스토리

학습 목표

1.1 애자일 소프트웨어 개발 기본

FA-1.1.1 (K1) 애자일 선언(Agile Manifesto)을 기반으로 하는 애자일 소프트웨어 개발의 기본 개념을 상기한다

FA-1.1.2 (K2) 전체 팀 접근법의 장점을 이해한다

FA-1.1.3 (K2) 빠르고 잦은 피드백의 장점을 이해한다

1.2 애자일 접근법

FA-1.2.1 (K1) 애자일 소프트웨어 개발 접근법을 상기한다

FA-1.2.2 (K3) 개발자와 업무 대표자*가 테스트할 사용자 스토리를 함께 작성한다

FA-1.2.3 (K2) 회고를 통해 애자일 프로젝트를 개선할 수 있음을 이해한다

FA-1.2.4 (K2) 지속적인 통합의 목적과 방법을 이해한다

FA-1.2.5 (K1) 반복주기(Iteration)와 출시(Release) 계획의 차이를 알고, 각각의 활동에 테스터가 어떻게 기여하는지 이해한다

** 업무 대표자(Business representatives) 역할에 대한 설명은 1 장 마지막 페이지 참조*

1.1 애자일 소프트웨어 개발 기본

애자일 프로젝트에 참여하는 테스터는 전통적인 프로젝트와는 다른 방식으로 일을 하게 된다. 테스터는 애자일 프로젝트를 뒷받침하는 가치와 원칙을 이해하고 개발자, 업무 대표자와 함께 전체 팀의 일원임을 깨달아야 한다. 애자일 프로젝트의 멤버는 서로 자주 소통해야 하며, 이를 통해 결함을 조기에 제거하고 제품의 품질을 향상시킬 수 있다.

1.1.1 애자일 소프트웨어 개발과 애자일 선언

2001 년에 널리 사용되던 경량 소프트웨어 개발 접근법을 대표하는 사람들이 모여 애자일 소프트웨어 개발을 위한 선언 또는 애자일 선언으로 알려진 공통적인 가치와 원칙에 동의하였다. 애자일 선언은 궁극적으로 추구하는 가치를 다음과 같은 4 가지 선언으로 나타냈다.

- 프로세스와 도구보다 개인과 그 개인들간의 상호작용을 중시한다
- 완벽한 문서보다 작동하는 소프트웨어를 중시한다
- 계약의 협상보다 고객과의 협업을 중시한다
- 계획을 고수하기보다 변화에 대한 대응을 중시한다

이 말은, 왼쪽에 있는 것들도 가치가 있지만, 우리는 오른쪽에 있는 것들에 더 높은 가치를 둔다는 뜻이다.

개인과 상호작용

애자일 개발은 매우 인간 중심적이다. 팀은 소프트웨어를 만드는 사람들로 구성되어 있으며, 도구나 프로세스에 의존하기 보다 지속적인 의사소통과 상호작용을 통해 더욱 효과적으로 일할 수 있다.

작동하는 소프트웨어

고객의 입장에서 보면 지나치게 자세한 설명서 보다 작동하는 소프트웨어가 훨씬 더 유용하고 가치 있으며, 이를 통해 개발팀은 신속한 피드백을 제공받을 수 있는 기회를 얻는다. 비록 기능이 축소되어 있으나 제대로 작동하는 소프트웨어는 개발 수명주기 초기에도 완전하게 동작하므로 애자일 개발은 타임 투 마켓 전략에 이점을 제공할 수 있다. 따라서 애자일 개발은 문제와 솔루션이 불분명하거나 새로운 영역에서 혁신을 원하는 등의 급격하게 변하는 비즈니스 환경에 특히 유용하다.

고객과의 협력

고객은 그들이 필요로 하는 시스템을 명세화 할 때 큰 어려움을 겪는다. 고객과 직접 협력하여 함께 일하는 것은 고객의 요구사항을 정확하게 이해할 가능성을 향상시킨다. 고객과의 계약도 중요하지만 아울러 그들과 함께 정기적으로 긴밀하게 협력해서 작업하는 것이 프로젝트의 성공 가능성을 더욱 높여준다.

변화 대응

변화는 소프트웨어 프로젝트에서 불가피하다. 사업 운영 환경, 법률, 경쟁사의 활동, 기술진보 및 다른 여러 요인들은 프로젝트와 프로젝트의 목적에 중요한 영향을 미칠 수 있다. 이러한 요소들은 개발 과정에서 수용되어야 한다. 이와 같이 변화를 포용하는 작업 관행에서의 유연성을 갖는 것은 단순히 계획을 엄격하게 준수하는 것보다 더 중요하다.

원칙

애자일 선언의 핵심적인 가치는 다음 **12 가지 원칙**에 잘 나타나 있다:

- 우리의 최우선 순위는 고객을 만족시키는 것이며, 이를 위해 조기에 지속적으로 가치 있는 소프트웨어를 전달한다
- 비록 개발의 후반부일지라도 요구사항 변경을 환영하라. 애자일 프로세스들은 변화를 통해 고객의 경쟁력에 도움을 준다
- 작동하는 소프트웨어를 자주 전달하라. 2 주 ~ 2 개월 사이에서 전달하되 그 간격이 짧을수록 좋다
- 비즈니스 쪽의 사람들과 개발자들은 프로젝트 전체에 걸쳐 날마다 함께 일해야 한다
- 동기가 부여된 개인들 중심으로 프로젝트를 구성하라. 그들이 필요로 하는 환경과 지원을 주고 그들이 일을 끝내리라고 신뢰하라
- 개발 팀 외부, 혹은 개발팀 내에서 정보를 가장 효율적이고 효과적으로 전달하는 방법은 직접 얼굴을 맞대고 대화하는 것이다
- 작동하는 소프트웨어를 태스크 진척의 최우선 지표로 삼는다
- 애자일 프로세스들은 지속 가능한 개발을 장려한다. 스폰서, 개발자, 사용자는 일정한 속도를 계속 유지할 수 있어야 한다
- 기술적 탁월성과 좋은 설계에 대한 지속적 관심이 기민함을 높인다
- 단순성이, 즉, 안 하는 일의 양을 최대화하는 기술이 창발(創發)한다
- 최고의 아키텍처, 요구사항, 설계는 자기 조직적인 팀으로부터 시작한다
- 팀은 정기적으로 어떻게 더 효과적이 될지 숙고하고, 이에 따라 팀의 행동을 조율하고 조정한다

다양한 애자일 접근법들이 이러한 가치와 원칙을 실행으로 옮길 수 있는 규정된 실천방안을 제공한다.

1.1.2 전체 팀 접근법

전체 팀 접근법이란 프로젝트의 성공을 보장할 수 있는 지식과 기술을 가진 모든 사람이 필수적으로 팀에 포함되어야 함을 의미한다. 팀에는 고객을 대표할 수 있는 사람과 고객 이외의 비즈니스 이해관계자들이 포함되어 이들이 함께 제품의 기능을 결정한다. 팀은 상대적으로 작게 구성되어야 하는데, 관찰에 따르면 성공적인 팀은 주로 적게는 3 명에서 많게는 9 명의 인원으로 구성된다. 같은 공간을 공유하는 것은 의사소통과 상호작용을 강하게 하므로 전체 팀이 동일한 작업 공간을 공유하는 것이 이상적이다.

전체 팀 접근법을 효과적으로 지원하기 위한 방법으로 일일 스탠드업 미팅(2.2.1 절 참조)을 실시하는데, 이 미팅에는 팀의 모든 멤버가 참여하여 현재 태스크 진척을 공유하고, 태스크 진행에 장애가 되는 모든 요소들을 확인한다.

작업 진행 상황을 공유하고 모든 문제점을 공유하는 일일 스탠드 업 미팅(2.2.1 절 참조)을 통해 전체 팀 접근법은 지원을 받는다. 전체 팀 접근법은 보다 효과적이고 효율적으로 팀의 역동성을 촉진한다.

제품 개발에 전체 팀 접근법을 사용하는 것은 애자일 개발의 주요 장점 중 하나이다. 전체 팀 접근법을 통해 다음과 같은 이익을 얻을 수 있다:

- 팀 내 의사 소통과 협력을 강화한다
- 팀 내의 다양한 스킬 셋을 활용해 프로젝트에 기여할 수 있다
- 제품의 품질을 모두의 책임으로 인식한다

애자일 프로젝트에서 품질에 대한 책임은 전체 팀에게 있다. 전체 팀 접근법의 핵심은 개발 과정의 모든 단계에 테스터, 개발자, 업무 대표자가 함께 참여해서 일한다는 것이다. 테스터는 원하는 품질 수준에 최대한 도달하기 위해 개발자와 업무 대표자 모두와 긴밀하게 일을 해야 한다. 이를 위해 업무 대표자가 적절한 인수 테스트를 만들 수 있도록 지원하고 협력하며, 개발자들이 테스트 전략에 동의할 수 있도록 협업해야 한다. 또한 테스트 자동화 접근 방법을 결정하기도 해야 한다. 이렇게 함으로써 테스터는 다른 팀 멤버에게 테스트 지식을 전달하고 확장시키며, 제품 개발에 영향을 미칠 수 있다.

전체 팀은 제품 기능을 제시하고, 분석하거나 추정하는 모든 협의나 회의에 참여한다. 모든 기능 토론에 테스터, 개발자, 업무 대표자가 참여하는 개념을 “3 의 힘(The Power of Three)”이라고 한다 [Crispin08].

1.1.3 조기에 지속적으로 얻는 피드백

애자일 프로젝트는 반복주기를 가능한 짧게 설정하여 개발 수명주기 전반에 걸쳐 프로젝트 팀이 제품 품질에 대한 피드백을 조기에 지속적으로 받을 수 있게 한다. 이를 위한 방법의 한 가지로 지속적인 통합을 사용할 수 있다 (1.2.4 절 참조).

순차 개발 접근법을 사용하는 경우, 고객은 프로젝트가 거의 완료된 단계가 되어서야 제품을 확인할 수 있으며, 이 시점에 고객에게 문제점이 발생하는 경우 개발팀에게 이를 해결할 충분한 시간이 주어지지 않게 된다. 프로젝트가 진행됨에 따라 지속적으로 고객의 피드백을 받음으로써 애자일 팀은 제품 개발 프로세스에 가장 새로운 변화를 통합할 수 있다. 조기에 지속적으로 얻는 피드백을 통해 팀은 가장 높은 비즈니스 가치를 갖는 기능이나 위험에 집중하고, 이를 또한 가장 먼저 고객에게 전달할 수 있게 된다. 아울러 팀의 역량 즉, '우리가 스프린트나 반복주기 동안 얼마나 많은 일을 할 수 있는가? 우리가 더 빠르게 일을 하는데 도움을 줄 수 있는 것은 무엇인가? 우리가 빠르게 일을 하는데 방해가 되는 요소는 무엇인가?' 등을 모든 사람에게 투명하게 내보임으로써 팀을 더 잘 관리하는 데 도움을 준다.

조기에 얻는 지속적인 피드백을 통해 얻을 수 있는 장점은 다음과 같다.

- 요구사항에 대한 잘못된 이해를 방지할 수 있다. 요구사항을 정확하기 이해하지 못해서 발생하는 결함은 개발 수명주기 막바지까지도 발견하지 못할 수 있으며 이를 수정하는 데는 더 많은 비용이 소요된다.
- 고객의 기능 요구를 명확히 하여 제품 개발 초기에 고객이 사용해볼 수 있도록 한다. 이를 통해 고객이 원하는 것을 제품에 더 명확히 반영할 수 있다.
- (지속적인 통합을 통해) 품질문제를 조기에 발견하고 분리시키며 해결할 수 있다.
- 애자일 팀의 생산성 및 배포 능력과 관련된 정보들을 전달한다.
- 프로젝트를 일관성 있게 추진하도록 한다.

1.2 애자일 접근법

조직에 따라 적용할 수 있는 애자일 접근법은 다양하다. 대부분의 애자일 조직에서 사용하는 실천방법으로는 사용자 스토리 도출, 회고, 지속적인 통합, 개별 반복주기 및 전체 출시 계획 등이 있다. 이 섹션에서는 일부 애자일 접근법에 대해 설명한다.

1.2.1 애자일 소프트웨어 개발 접근법

많은 애자일 접근법들이 존재하며, 각각의 접근법들은 서로 다른 방법으로 애자일 선언의 가치와 원칙들을 구현한다. 이 실라버스에서는 애자일 방식을 대표하는 익스트림 프로그래밍(XP), 스크럼과 칸반(Kanban) 세가지 접근법을 소개한다.

익스트림 프로그래밍

켄트 벡이 처음 시작한 익스트림 프로그래밍은 애자일 접근법 중 하나로 소프트웨어 개발을 특정한 가치, 원칙 및 개발 실천법으로 설명한다.

XP 는 다섯 가지 가치 즉, 의사소통, 단순성, 피드백, 용기 및 존중을 통해 개발을 가이드 한다.

XP 는 또한 추가적인 가이드라인으로 일련의 원칙들을 제공하는데, 그 원칙들은 다음과 같다: 인간성, 경제성, 상호 이익, 자기 유사성, 개선, 다양성, 반성, 몰입, 기회, 중복, 실패, 품질, 잔걸음 그리고 수용된 책임감

XP 는 13 가지 실천법을 제시한다: 함께 앉기, 전체 팀, 정보를 제공하는 작업 공간, 열정적인 작업, 짝 프로그래밍, 스토리, 주 단위 주기, 사분기 주기, 여유, 10 분 빌드, 지속적인 통합, 테스트 우선 프로그래밍, 점진적 설계

오늘날 사용되는 애자일 소프트웨어 개발 방식의 대부분은 XP 및 XP 의 가치와 원칙에 영향을 받았다. 예를 들어, 애자일 팀은 스크럼에 XP 실천법을 자주 포함시킨다.

스크럼 Scrum

스크럼은 애자일 관리 프레임워크의 하나로 다음 구성요소와 실천법들을 포함하고 있다 [Schwaber01]:

- 스프린트: 스크럼은 프로젝트를 고정된 길이(보통 2~4 주)의 주기로 나눈다. 이 주기를 스프린트라고 부른다.
- 증분된 산출물: 모든 스프린트는 잠재적으로 출시/배포 가능한 제품(점진적 개발)을 산출한다.
- 제품 백로그: 제품 오너는 계획된 제품 아이템의 우선 순위 목록(제품 백로그)을 관리하며, 이 목록은 스프린트가 진행됨에 따라 계속 진화한다(백로그 정제).
- 스프린트 백로그: 스프린트 시작 시점에 스크럼 팀은 제품 백로그에서 우선 순위가 가장 높은 항목 집합(스프린트 백로그)을 선택한다. 제품 오너의 강제에 의하지 않고 스크럼 팀에서 스프린트 내에 구현 가능한 항목을 선택하므로 밀기(Push) 원리가 아닌 당김(Pull) 원리를 따른다.

- 완료의 정의: 각 스프린트 완료 시점에서 잠정적으로 출시 가능한 제품이 산출되었는지를 확인하기 위해, 스크럼 팀은 적절한 스프린트 종료 기준을 협의하고 정의한다. 종료 기준 논의를 통해 애자일 팀은 백로그 아이템과 제품 요구사항을 더 깊이 이해하게 된다.
- 타임 박싱: 스프린트를 진행하는 애자일 팀이 해당 스프린트 내에서 완료할 수 있을 것으로 예상하는 작업, 요구사항 혹은 기능들만이 해당 스프린트 백로그의 일부가 된다. 만약 개발 팀이 스프린트 내에서 작업을 완료할 수 없다면, 관련 제품의 기능은 스프린트에서 제거되고 작업은 다시 제품 백로그로 이동된다. 타임 박싱은 다른 상황의 작업에도 적용된다. (예: 회의 시작 및 종료 시간)
- 투명성: 개발팀은 일일 스크럼이라 불리는 회의에서 매일 스프린트 상태를 공유한다. 일일 스크럼 회의를 통해 테스트 결과를 포함해 현재 스프린트의 내용과 진척을 팀, 관리자 및 모든 관련담당자에게 공유한다.

스크럼에서는 다음과 같은 3 가지 역할을 정의한다:

- 스크럼 마스터: 스크럼 원칙과 규칙이 구현되고 준수되게 한다. 팀이 원칙과 규칙을 따르지 못하게 하는 모든 요소들, 자원 문제 또는 기타 장애물을 해결한다. 이 사람은 팀 리더가 아닌 코치이다.
- 제품 오너: 고객을 대표하고 제품 백로그를 생성, 유지하고 우선순위를 정한다. 이 사람은 팀 리더가 아니다.
- 개발팀: 제품을 개발하고 테스트한다. 개발팀은 팀 리더 없이 스스로 결정하는 자기 조직화 된 팀이다. 또한 이 팀은 복합 기능 팀이다 (2.3.2 절과 3.1.4 절 참조).

XP 와 달리 스크럼은 특정 소프트웨어 개발 기술(예: 테스트 우선 프로그래밍)을 강요하지 않는다. 또한 스크럼은 스크럼 프로젝트에서 테스트가 어떻게 수행되어야 하는지에 대한 지침을 제공하지 않는다.

칸반 Kanban

칸반은 관리 접근법의 하나로 애자일 프로젝트에서 종종 사용된다. 일반적인 목적은 가치 사슬 내의 태스크 흐름을 시각화하고 최적화하는 것이다. 칸반은 세 가지 도구를 사용한다 [Linz14]:

- 칸반 보드: 관리되어야 할 가치 사슬은 칸반 보드를 통해 가시화 된다. 각 열은 관련된 일련의 활동 단계(예: 개발 또는 테스트)를 보여준다. 제작되는 아아템이나 처리할 태스크는 왼쪽에서 오른쪽으로 단계별로 이동하는 카드로 표현된다.
- 진행 작업 제한: 병렬로 동시에 실행되는 태스크의 양은 엄격히 제한된다. 단계 및 보드 전체에서 허용 가능한 카드의 최대수를 제어하는 방식을 사용한다. 작업자는 단계별로 여유 능력이 있을 때마다 이전 단계의 카드를 가져 온다.

- 리드 타임: 칸반은 완전한 가치 흐름에 대해 (평균) 리드 타임을 최소화함으로써 작업의 연속적인 흐름을 최적화하기 위해 사용된다.

칸반과 스크럼은 일부 유사성을 가진다. 두 프레임워크는 진행중인 작업의 시각화(예: 공동의 화이트보드)를 통해 작업 내용과 진척 상황에 대한 투명성을 제공한다. 예약되지 않은 작업은 백로그에 대기하고 있다가 새로운 공간(생산 능력)을 사용할 수 있을 때 칸반 보드로 이동한다.

칸반에서의 반복주기나 스프린트는 선택 사항이다. 칸반 프로세스에서는 산출물을 출시 단위가 아니라 항목별로 릴리즈 할 수 있다. 따라서 동기화 메커니즘인 타임 박싱은 칸반에서 선택적으로 적용된다. 반면 스프린트 내에서 모든 태스크를 동기화하는 스크럼에서는 타임 박싱이 필수적으로 적용된다.

1.2.2 사용자 스토리 공동 작업

충분치 않은 명세는 프로젝트의 실패를 야기하는 주요한 원인이다. 사용자의 실제 요구사항에 대한 스스로의 통찰력 부족, 대상 시스템에 대한 전체적인 비전의 부재, 중복되거나 모순되는 기능 및 기타 커뮤니케이션 이슈로 명세와 관련된 문제들이 발생할 수 있다. 애자일 개발환경에서 사용자 스토리는 개발자, 테스터 및 업무 대표자 관점의 요구사항들을 모두 수집하여 작성한다. 기능의 비전 공유는 순차적 개발 모델에서는 요구사항이 기록된 후 공식 리뷰를 통해 수행되지만, 애자일 개발환경에서는 요구사항이 작성되는 동안 비공식적인 리뷰를 통해 수행된다.

사용자 스토리는 기능 및 비기능 특성을 모두 해결해야 한다. 각각의 사용자 스토리는 이러한 특성에 대한 인수 기준을 가지고 있어야 한다. 이러한 조건은 업무 대표자, 개발자와 테스터 간의 협의에 의해 정의되어야 한다. 인수 기준은 개발자와 테스터에게 기능의 확장된 비전을 제공하고 업무 대표자는 유효성을 검사하기 위한 기준을 제공한다. 애자일 팀은 인수 기준이 충족되었을 때 작업이 완료된 것으로 간주한다.

일반적으로 테스터가 다른 이해관계자들과 다른 고유한 관점, 가령, 누락된 세부 사항 및 비기능적 요구사항 식별 등을 통해 사용자 스토리를 향상시킨다. 또한 테스터는 업무 대표자에게 사용자 스토리와 관련된 개방형 질문을 하고, 테스트 방법을 제안하며 인수조건을 확인함으로써 사용자 스토리 향상에 기여하기도 한다.

사용자 스토리의 공동 저자는 브레인 스토밍과 마인드 매핑 등의 기술을 사용할 수 있다. 테스터는 INVEST 기법을 사용할 수 있다[INVEST]:

- 독립적이다
- 협상 가능하다
- 사용자와 고객에게 가치가 있다

- 추정 가능하다
- 작다
- 테스트 가능하다

3C 개념[Jeffries00]에 따르면 사용자 스토리는 다음 세 가지 요소의 결합이다:

- **카드**: 카드는 사용자 스토리를 기록한 물리적 매체로, 요구사항, 요구사항의 중요도, 예상되는 개발 및 테스트 기간, 인수 기준을 기록한다. 상세 설명은 향후 제품 백로그에 사용되므로 정확하게 기술해야 한다.
- **대화**: 대화는 소프트웨어를 어떻게 사용하는지에 대해 설명한다. 대화는 문서 또는 구두로 할 수 있다. 개발자와 업무 대표자와 다른 관점을 갖는 테스터[ISTQB_FL_SYL]는 생각, 의견, 경험의 교환에 있어 가치 있는 의견을 제공한다. 대화는 출시 계획 단계에서 시작하여 스토리의 일정이 추정 될 때까지 계속된다.
- **확인**: 대화에서 논의된 인수 기준을 사용해 스토리가 완료되었는지 확인한다. 이러한 인수 기준은 여러 사용자 스토리에 동시에 적용될 수도 있다. 인수 기준 만족 여부를 확인하기 위해 긍정적 테스트와 부정적 테스트를 모두 사용해야 한다. 확인 과정에는 다양한 참가자가 테스터의 역할을 수행한다. 참가자에는 개발자는 물론 성능, 보안, 상호 운용성 및 다른 품질 특성에 대한 전문가들도 포함된다. 정의된 인수 기준이 테스트 되고 만족되었음이 증명되면 해당 스토리는 완료된 것으로 확인한다.

애자일 팀은 다양한 방법으로 사용자 스토리를 작성한다. 사용자 스토리를 문서화하는 방법에 관계없이 문서는 간결하고 충분히 필요한 내용이 들어가 있어야 한다.

1.2.3 회고

애자일 개발에서 회고는 각 반복주기의 가장 마지막에 수행하는 회의로 '이번 반복주기에서 무엇이 성공적이었는지', '다음 반복주기에서 무엇을 향상시킬 수 있는지', '어떻게 성공을 유지하고 개선 사항을 통합할 수 있을지'에 대해 논의한다. 주로 프로세스, 사람, 조직, 관계, 도구와 같은 주제를 다룬다. 적절한 후속 활동을 이끌어내는 회고 모임은 조직의 개발 및 테스트의 지속적인 개선에 매우 중요하다.

회고를 통해 테스트와 연관된 개선 관련 의사 결정을 할 수 있으며 여기에는 테스트 효과성, 테스트 생산성, 테스트 케이스 품질 및 팀 만족 여부 등이 포함된다. 또한 응용 프로그램, 사용자 스토리, 특성 또는 시스템 인터페이스의 테스트 용이성을 해결할 수 있다. 결함 근본 원인 분석을 통해 테스팅과 개발의 개선을 추진할 수 있으며, 일반적으로 팀은 반복주기를 거듭하며 조금씩 개선을 진행함으로써 꾸준한 속도로 개선을 수행할 수 있게 된다.

회고의 시기와 구성은 애자일 기법에 따라 달라진다. 중재자(Facilitator)가 회의를 구성하고 진행하는 동안 업무 대표자 및 팀 참가자는 각 회고에 참석한다. 어떤 경우에는 팀 미팅에 다른 참가자를 초대할 수 있다.

테스터는 회고에서 중요한 역할을 수행한다. 테스터는 팀에 소속되어 있으며 ISTQB 실라버스 1.5 절에 언급된 바와 같이 그만의 독특한 관점을 팀에 제공한다. 모든 스프린트에서 테스팅을 수행하고 프로젝트의 성공에 실질적으로 기여해야 한다.
테스터를 포함한 모든 팀 구성원은 테스팅을 포함한 모든 활동에 입력 정보를 제공할 수 있다.

회고는 상호 신뢰를 특징으로 하는 전문적인 환경에서 수행되어야 한다. 성공적인 회고를 위해서는 다른 리뷰들과 동일한 특성들을 참고해야 하며, 이런 특성에 관한 내용은 ISTQB Foundation 실라버스 3.2 절을 참조한다.

1.2.4 지속적인 통합

증분 산출물을 전달하기 위해서 모든 스프린트가 완료되는 시점에는 신뢰할 수 있고 제대로 작동하는 통합된 소프트웨어가 완성되어야 한다. 지속적인 통합은 적어도 하루에 한번 소프트웨어의 모든 변경 사항을 통합하고 정기적으로 변경된 모든 구성 요소를 통합함으로써 이 문제를 해결한다. 형상 관리, 편집, 소프트웨어 빌드, 배포 및 테스트를 하나의 자동화된 반복적인 프로세스로 통합한다. 개발자는 자신의 작업을 지속적으로 통합하고, 지속적으로 구축하고, 지속적으로 테스트하기 때문에 코드의 결함을 더 빨리 발견할 수 있다.

개발자가 코딩, 디버깅 후 공유 소스 코드 저장소에 코드를 체크인하고 나면 지속적인 통합 프로세스는 다음과 같은 활동을 자동으로 수행한다:

- 정적 코드 분석: 정적 코드 분석을 실행하고 결과를 보고한다
- 컴파일: 코드를 컴파일하고 링크를 걸어 실행 파일을 생성한다
- 단위 테스트: 단위 테스트를 실행하고 코드 커버리지를 확인하며, 테스트 결과를 보고한다
- 배포: 테스트 환경에 빌드를 설치한다
- 통합 테스트: 통합 테스트 실행과 경과를 보고한다
- 보고(현황판): 모든 활동의 상태는 오픈된 장소에 공개하거나, 해당 팀에 메일을 보내 상태를 공유한다

자동화 빌드 및 테스트 프로세스가 매일 실행되고 초기에 빠르게 통합 에러를 발견한다. 지속적인 통합 환경에서 애자일 테스터는 자동화된 테스트를 정기적으로 수행하며(자동화 테스트의 어느 부분은 지속적인 통합 환경에서 자체적으로 제공하기도 한다), 코드 품질에 대한 즉각적인 피드백을 팀에 제공할

수 있다. 이러한 테스트 결과는 특히 자동화된 보고서가 프로세스에 통합되어 있을 때 모든 팀 구성원에게 공개된다. 반복주기가 진행되는 동안 자동화된 리그레션 테스트를 지속적으로 수행할 수 있고, 바람직하게 자동화된 리그레션 테스트는 이전 반복주기에서 전달된 사용자 스토리를 포함하여 가능한 많은 기능을 포함한다. 자동화된 리그레션 테스트에서 좋은 커버리지는 큰 규모의 통합 시스템을 구축하고 테스트를 수행하는데 도움이 된다. 리그레션 테스트를 자동화함으로써 애자일 테스터들은 새로운 기능과 변경된 구현 사항에 대한 수동 테스트 및 결함 수정 확인 테스트에 집중할 수 있다.

일반적으로 지속적인 통합을 사용하는 조직은 테스트를 자동화하는 것뿐만 아니라 지속적인 품질 관리를 위해 빌드 도구를 사용한다. 즉, 단위 및 통합 테스트를 실행하는 것 외에도 이러한 도구는 추가적인 정적 및 동적 테스트, 성능 측정 및 프로파일링, 소스코드 문서화를 실행하고 수동 품질 보증 프로세스를 용이하게 할 수 있다. 이러한 응용 프로그램의 지속적인 품질 관리는 제품의 품질을 향상시킬 뿐만 아니라 모든 개발의 완료 후에야 품질 관리를 적용하는 전통적인 방법을 대체하여 제품 출시에 걸리는 시간을 단축시킨다.

빌드 도구는 자동 배포 도구에 연결하여 사용할 수 있다. 자동 배포 도구들은 지속적인 통합 서버 혹은 빌드 서버에서 적절한 빌드 산출물을 복사하여, 하나 혹은 그 이상의 개발 환경, 테스트 환경, 스테이징 환경 혹은 실제 사용환경에 배포한다. 이것은 해당 환경에서 출시를 담당하는 전문직원 또는 프로그래머와 관련된 오류 및 지연을 줄일 수 있게 해준다.

지속적인 통합을 통해 얻을 수 있는 이점은 다음과 같다:

- 문제와 변경관련 충돌의 조기발견과 근본적인 원인 분석이 가능하다
- 개발팀에 코드가 작동하는지에 대한 정기적인 피드백을 제공한다
- 개발 중인 소프트웨어 버전 당일 내에 테스트 중인 버전을 유지한다
- 코드 베이스가 변경되는 즉시 (자동화 된) 재테스팅을 수행함으로써 개발자의 코드 리팩토링과 관련된 리그레션 리스크를 줄일 수 있다
- 탄탄한 기초를 기반(빌드 및 테스트가 정상 수행되지 않으면 해당 이슈를 먼저 해결해야 한다)으로 코드 개발 작업에 대한 확신을 제공한다
- 제품의 추가 개발에 대한 완료 여부를 시각화 함(테스트 주도 개발 방법론을 도입한 경우, 먼저 작성된 테스트 코드를 지속적인 통합 환경에서 수행하게 된다. 자동화된 단위 테스트가 모두 성공할 때까지 제품 코드를 작성하게 된다)으로써 개발자와 테스터를 격려한다
- 빅뱅 통합 방법에 따른 일정 상의 리스크를 제거한다
- 실행 가능한 소프트웨어를 지속적으로 적용함으로써 이 소프트웨어는 스프린트 테스팅, 데모, 교육 목적 등으로 활용할 수 있다
- 반복적인 수동 테스트 작업을 줄일 수 있다

- 품질 및 테스트를 개선하기 위해 내린 결정에 대한 빠른 피드백을 제공한다.

그러나, 지속적인 통합에도 다음과 같은 리스크와 어려움이 존재한다:

- 지속적인 통합 도구가 도입되고 잘 유지되어야 한다
- 지속적인 통합 프로세스를 정의하고 설정해야 한다
- 테스트 자동화는 추가적인 자원을 필요로 하며 설정이 복잡할 수 있다
- 자동화된 테스트의 장점을 활용하기 위해서는 철저한 테스트 커버리지를 만족시켜야 한다
- 팀이 단위 테스트에 지나치게 의존한 나머지 시스템/인수 테스팅을 불충분하게 수행할 수 있다

지속적인 통합 환경을 잘 구축하기 위해서는 테스팅 도구, 빌드 프로세스 자동화 도구, 형상 관리 도구 등을 함께 사용해야 한다.

1.2.5 출시와 반복주기 계획

ISTQB Foundation 실라버스[ISTQB_FL_SYL]에서 언급한 바와 같이, 계획은 지속적인 활동이며, 이것은 애자일 수명주기에서도 마찬가지이다. 애자일 라이프사이클의 경우 계획 활동에는 출시 계획과 반복주기 계획 두 가지가 있다.

출시 계획은 제품의 출시를 계획하며, 종종 프로젝트 시작 몇 개월 전에 수립된다. 출시 계획 단계에서는 새로운 백로그를 정의하거나 기존 백로그를 재정의하는 활동을 실행하며 이 과정에서 작은 사용자 스토리들을 모아 큰 사용자 스토리로 만들기도 한다. 출시 계획은 모든 반복주기에 대한 테스트 접근법과 테스트 계획을 위한 기초를 제공한다. 출시 계획은 상위 레벨 수준의 계획이다.

출시 계획에서 업무 대표자는 팀과 공동으로 출시에 대한 사용자 스토리를 설정하고 우선순위를 정한다. (1.2.2 절 참조) 이 사용자 스토리에 기반하여 프로젝트 리스크와 품질 리스크를 식별하고 상위 레벨 수준에서 공수를 추정한다 (3.2 절 참조).

테스터도 출시 계획에 참여하며 특히 다음과 같은 활동을 통해 계획에 기여할 수 있다:

- 테스트 가능한 사용자 스토리를 정의한다. 여기에는 인수 기준도 포함된다
- 프로젝트 리스크 및 품질 리스크 분석에 참여한다
- 사용자 스토리와 관련된 테스트 공수를 추정한다
- 필요한 테스트 레벨을 정의한다
- 출시에 대한 테스팅을 계획한다

출시 계획이 완료되면 첫 번째 반복주기를 위한 계획을 수립한다. 반복주기 계획은 하나의 반복주기의 종료 조건과 반복주기 백로그에 관한 내용을 다룬다.

반복주기 계획에서 팀은 출시 백로그로부터 사용자 스토리를 선택하고 사용자 스토리를 상세화하고, 사용자 스토리를 위한 리스크 분석을 실행하고, 각 사용자 스토리에 필요한 작업을 추정한다. 선택한 사용자 스토리가 너무 모호하여 명확화할 수 없는 경우, 애자일 팀은 해당 스토리를 거부하고 우선 순위에 따라 다음 사용자 스토리를 선택할 수 있다. 업무 대표자는 각 스토리에 대한 팀의 질문에 응답해야 하며, 이를 통해 팀은 각 스토리와 관련해 무엇을 구현하고 어떻게 테스트해야 할지 이해하게 된다.

팀의 속도와 선택되고 추정된 사용자 스토리의 크기에 따라 반복주기에서 구현되는 스토리의 수는 달라진다. 반복주기 내용이 확정된 후 사용자 스토리는 작업으로 분할되고 적절한 팀 멤버에게 할당된다.

테스터 역시 반복주기 계획에 참가하며 특히 다음과 같은 활동을 통해 계획 단계에 기여할 수 있다:

- 사용자 스토리와 관련된 상세 리스크 분석에 참여한다
- 사용자 스토리의 테스트 용이성을 결정한다
- 사용자 스토리와 관련된 인수 테스트를 작성한다
- 사용자 스토리를 태스크 단위(특히 테스팅 업무 단위)로 분리한다
- 모든 테스팅 업무와 관련된 공수를 추정한다
- 테스트 대상 시스템의 기능적/비기능적 측면을 식별한다
- 다양한 테스트 레벨의 테스팅 자동화를 지원하고 이에 참여한다

출시 계획은 프로젝트 진행에 따라 변경될 수 있으며, 여기엔 제품 백로그에 포함된 개별 사용자 스토리의 변경도 포함될 수 있다. 이러한 변경은 내부 또는 외부 요인에 의해 유발될 수 있다. 내부 요인으로는 배포 능력, 속도, 기술적인 문제 등이 포함될 수 있으며 외부 요인으로는 새로운 시작과 기회, 새로운 경쟁 업체 발견 혹은 비즈니스 위협에 따른 출시 목표나 출시 날짜 변경이 있다. 또한 반복주기 계획은 반복주기 중에 변경될 수 있다. 예를 들어, 비교적 간단하다고 간주했던 특정 사용자 스토리가 분석 결과 예상보다 더 복잡한 경우 등이다.

이러한 변화는 테스터에게 도전이 될 수 있다. 테스터는 테스트 계획을 위해 출시의 큰 그림을 이해해야 하며, ISTQB Foundation 실라버스[ISTQB_FL_SYL] 1.4 절에 설명된 대로 테스트 개발을 위해 각 반복주기에서 적절한 테스트 베이시스와 테스트 오라클을 가지고 있어야 한다. 필요한 정보는 개발 초기에 테스터가 사용 가능해야 하고 일어날 변화는 애자일 원칙에 따라 수용해야 한다. 이러한 딜레마는 테스트 전략과 테스트 문서에 관한 주의 깊은 결정을 필요로 한다. 애자일 테스팅 변화에 대한 자세한 내용은 [Black09] 제 12 장을 참조한다.

출시와 반복주기 계획은 개발 계획 활동에 따른 테스트 계획을 다룬다. 고려해야 할 특정 테스트 관련 문제는 다음과 같다:

- 테스팅의 목적, 목적에 따른 테스팅의 범위, 테스트 목표 및 이러한 결정에 대한 이유
- 테스트 활동을 수행할 팀 멤버
- 필요한 테스트 데이터와 환경, 그리고 테스트 환경이나 데이터에 대한 어떠한 추가 또는 변경이 프로젝트 동안 발생할지에 대한 여부
- 개발 활동과 관련이 있거나 의존적인 테스트 활동을 포함한 기능 및 비기능 테스트 활동 및 이를 위한 타이밍, 순서, 종속성과 전제 초건(예: 다른 기능 또는 테스트 데이터에 따라 달라지는 기능에 대해 리그레션 테스트를 얼마나 자주 실행해야 하는가?)
- 해결해야 하는 프로젝트와 품질 리스크(3.2.1 절 참조)

아울러 큰 팀의 공수를 추정하는 경우에는 요구된 테스팅 활동의 완료에 필요한 시간과 공수를 포함시켜야 한다.

[참고 정보]

Business Representative(업무 대표자)

• Managing Business relationships and participate in strategy development, or system/application upgrades with stakeholders during concept through delivery

• Documenting the strategy, roadmap, business case, and oversight model in the execution of solutions assigned

• Conducting\leading high level solution sessions with IT and Business partners, initiate program through Banamex USA's Program Management Office, and communicating status throughout the program to all levels of the organization

• Collaborating with Product Managers to document and translate business strategy into technology roadmaps and capability matrixes in coordination with internal IT teams, internal/external vendors, Business, Operations, and other impacted areas to enforce a consistent strategy, roadmap, and standard documentation

• Actively drive simplification of technology environment while leveraging the necessary platforms where appropriate

• Directly impact business direction by influencing strategic\functional decisions through advice, counsel and guidance

• Working closely with the Business, Product Managers & Operations to respond to initiative work driven by regulatory or strategic needs

• Coordinating with IT Program/Project Managers and internal/external vendor(s) to ensure appropriate integration of functions to meet goals; define necessary system enhancements

• Tracking and communicating overall technology strategy and progress to the IT Program/Project Manager and all level of the organization as required

• Understanding overall Business and Information Technology strategic plans for supported applications and upcoming applications

• Escalating application risks to the IT Program/Project Manager when appropriate

2. 기본 애자일 테스팅 원칙, 예제 & 프로세스 – 105 분

키워드

빌드 베리피케이션 테스트, 형상 관리 아이템, 형상 관리

학습 목표

2.1 전통적인 테스팅과 애자일 접근법의 테스팅 차이

FA-2.1.1 (K2) 애자일 프로젝트와 다른 프로젝트의 테스트 활동의 차이점을 설명한다

FA-2.1.2 (K2) 개발 및 테스트 활동이 애자일 프로젝트에서 어떻게 통합되는지 설명한다

FA-2.1.3 (K2) 애자일 프로젝트의 독립적인 테스트 역할을 설명한다

2.2 애자일 프로젝트의 테스팅 상태

FA-2.2.1 (K2) 테스트 절차와 제품 품질을 포함하여 애자일 프로젝트에서 테스팅의 상태를 의사소통하는데 사용되는 도구와 기법을 설명하시오

FA-2.2.2 (K2) 다수의 반복주기에 걸쳐 테스트의 발전 과정을 설명하고, 테스트 자동화가 애자일 프로젝트에서 리그레션 리스크를 관리하는데 중요한 이유에 대해 설명하시오

2.3 애자일 팀 내 테스터의 역할과 기술 역량

FA-2.3.1 (K2) 애자일 팀에서 테스터의 역량(사람, 도메인 그리고 테스팅)을 이해한다

FA-2.3.2 (K2) 애자일 팀에서 테스터의 역할을 이해한다

2.1 전통적인 테스팅과 애자일 접근법의 테스팅 차이

ISTQB Foundation 실라버스[ISTQB_FL_SYL]와 [Black09]에서 설명한 바와 같이, 테스트 활동은 개발 활동과 연관되어 있고 수명주기에 따라 다양하게 변화한다. 테스터는 효과적이고 효율적인 작업을 위해 애자일 수명주기와 기존의 수명주기(예: V -모델과 같은 순차 모델 또는 RUP 와 같은 반복적 모델)의 차이점을 이해하고 있어야 한다. 애자일 모델은 테스팅과 개발 활동이 통합되어 있다는 점을 포함해 프로젝트 산출물, 용어, 다양한 테스트 레벨에서 사용되는 시작 및 종료 조건, 도구의 활용 및 독립적인 테스팅을 효과적으로 구현하는 데 있어서 기존 모델과 다르다.

테스터는 조직에 따라 수명주기 구현 방법이 상당히 다르다는 점을 명심해야 한다. 특정 조직의 수명주기 구성은 특별 주문(Customization)으로 제작된다. 다양한 실천방법을 적용함으로써 애자일 수명주기의 이상적인 형태가 달라질 수도 있다. 테스터는 해당 조직에서 사용하는 소프트웨어 개발 실천방법을 포함해 주어진 환경을 적절하게 적용할 수 있어야 한다. 그렇지 않으면 테스터로서 직무에 실패할 수 있다.

2.1.1 테스팅과 개발 활동

전통적인 수명주기와 애자일 수명주기의 주요 차이점 중 하나는 주기가 매우 짧은 반복이라는 개념이며, 각 반복주기가 완료되는 시점에는 이해관계자에게 가치를 전달할 주요 기능을 갖는 동작하는 소프트웨어가 만들어져야 한다. 프로젝트 시작 단계에서 출시 계획을 수립하며, 이후 일련의 반복주기가 진행된다. 각 반복주기의 시작 단계에서 반복주기 계획을 수립한다. 선택된 사용자 스토리가 개발되고 시스템에 통합되어 테스트가 실행된다. 이러한 반복주기는 각 반복주기마다 상당량의 병렬 및 오버랩(동시 발생)을 포함한 개발, 통합 그리고 테스팅 활동이 일어나는 매우 역동적인 활동으로 이루어진다. 테스팅 활동은 반복주기의 마지막 활동이 아니라 반복주기를 진행하는 동안 지속적으로 수행된다.

테스터, 개발자 그리고 비즈니스 이해관계자 모두 기존의 수명주기에서와 마찬가지로 테스팅에서 각자의 역할을 가지고 있다. 개발자는 사용자 스토리에서 기능을 개발하고 단위 테스트를 수행하며, 이후 테스터는 이러한 기능들을 테스트한다. 비즈니스 이해관계자도 구현기간 동안 스토리를 같이 테스트한다. 비즈니스 이해관계자는 작성된 테스트 케이스를 활용하기도 하고, 개발팀에게 빠른 피드백을 제공하기 위한 목적으로 기능을 사용해보기도 한다.

어떤 경우에는 잔존 결함 혹은 다른 형태의 기술 채무를 해결하기 위해 주기적으로 반복주기를 강화(Hardening)하거나 안정화(Stabilization)시키기도 한다. 그러나 가장 좋은 방법은 모든 기능이

시스템과 같이 테스트되고 통합되어야만 완료되었다고 간주하는 것이다 [Goucher09]. 또 다른 좋은 방법은 이전 반복주기에서 남겨진 결함들을 다음 반복주기 시작 시 모두 처리하는 것이다. 물론 결함들은 다음 반복주기의 백로그에 포함된다 (이것을 “결함 먼저 수정하기”라고 한다). 그러나 일각에서는 이러한 접근법을 사용하는 경우, 해당 반복주기에서 수행해야 할 태스크량을 전체적으로 확인할 수 없음은 물론, 남아 있는 기능들을 구현하는데 필요한 공수를 추정하기는 더 어렵다는 의견을 제시하기도 한다. 몇 차례의 반복주기가 완료된 시점에서 소프트웨어 인도(Delivery)를 위한 일련의 출시 활동이 발생할 수 있으며, 하나의 반복주기가 완료되는 시점마다 소프트웨어 인도가 발생할 수 있다.

테스트 전략의 하나로 리스크 기반 테스팅 전략을 사용하는 경우, 상위 레벨의 리스크 분석은 출시 계획 중에 수립하며, 대개 테스터들이 리스크 분석을 주도한다. 그러나, 각각의 반복주기와 관련된 특정한 품질 리스크는 반복주기 계획 단계에서 정의되고 평가된다. 리스크 분석 결과는 개발 순서는 물론, 구현된 기능과 관련된 테스팅의 우선 순위와 깊이에 영향을 미칠 수 있다. 또한 각 기능에 필요한 테스트 공수의 추정에 영향을 미친다(3.2 절 참조).

일부 애자일 방법론(예: 익스트림 프로그래밍)에서는 페어링을 사용한다. 페어링 기법에서는 두 명의 테스터가 짝을 지어 하나의 기능을 같이 테스트하며, 경우에 따라 개발자와 테스터가 협업을 하기도 한다. 테스트 팀이 분산되어 있는 경우에는 페어링을 적용하기 쉽지 않으나, 분산된 페어링을 수행하기 위해 적절한 프로세스나 도구들을 활용할 수 있다. 분산된 환경에서의 태스크 수행에 대한 자세한 내용은 [ISTQB_ALTM_SYL] 2.8 절을 참조한다.

테스터는 팀 내에서 테스팅 및 품질과 관련된 코칭 역할을 수행할 수 있다. 즉 테스팅 지식을 팀과 공유하고 팀의 품질 보증 태스크를 지원할 수 있으며, 이를 통해 제품 품질에 대한 공동의 책임을 독려한다.

많은 애자일 팀이 모든 테스팅 레벨에서의 테스트 자동화를 수행하고 있으며, 이는 테스터들이 자동화된 테스트 케이스와 수행 결과를 만들어내고, 실행하고, 모니터링하고 유지하는데 많은 시간을 사용해야 함을 의미한다. 자동화 테스트를 대량으로 수행하기 때문에, 애자일 프로젝트에서 수행되는 수동 테스팅의 대부분은 소프트웨어 공격, 탐색적 테스팅, 에러 추정과 같은 경험기반 혹은 결함기반 기법들이 사용된다 ([ISTQB_ALTA_SYL], 3.3 과 3.4 절 그리고 [ISTQB_FL_SYL], 4.5 절 참조).

개발자들은 단위 테스트 생성에 집중하며, 테스터는 자동화된 통합, 시스템, 시스템 통합 테스트를 만드는데 초점을 맞추어야 한다. 이런 이유로 애자일 팀은 높은 기술 및 테스트 자동화 경험을 가진 테스터를 선호하는 경향이 있다.

애자일 원칙의 한가지 핵심은 변화가 프로젝트 전반에 걸쳐 발생할 수 있다는 것이다. 따라서 애자일 프로젝트에서는 가벼운 제품 설명서를 선호한다. 기존 기능에 대한 변경은 테스팅 특히, 리그레션 테스팅에 영향을 미친다. 자동화된 테스팅의 사용은 변화와 관련된 테스트 노력의 양을 관리하는 하나의 방법이다. 그러나 변화의 양이 변화로 인해 발생 가능한 리스크에 대처할 수 있는 팀의 능력을 초과하지 않도록 주의해야 한다.

2.1.2 프로젝트 작업 산출물

애자일 테스터는 다음과 같은 세 가지 작업 산출물에 특히 관심을 가져야 한다:

1. 요구사항(예: 요구명세)과 사용 방법(예: 사용자 설명서)을 설명하는 비즈니스 중심의 산출물
2. 개발 작업 산출물: 시스템 설계(예: 데이터베이스 개체 관계 다이어그램), 실제 구현된 시스템(예: 코드) 또는 코드의 각 부분에 대한 평가(예: 자동화된 단위 테스트)
3. 테스트 작업 산출물: 시스템 테스트 방법(예: 테스트 전략과 계획), 실제 시스템 테스트 실행(예: 수동 및 자동화 테스트) 또는 테스트 결과 (예: 2.2.1 절에서 언급된 테스트 대시 보드)

일반적으로 애자일 프로젝트에서는 많은 양의 문서 생산을 지양하며 대신 작동하는 소프트웨어 및 해당 소프트웨어가 요구사항을 만족함을 입증하는 자동화된 테스트 코드에 집중한다. 고객에게 가치를 전달하지 못하는 문서는 과감히 줄일 수 있다. 애자일 프로젝트에서는 문서를 줄여 개발 효율성을 증가시키는 관점과 비즈니스, 테스팅, 개발 및 유지보수 활동을 지원하기에 충분한 문서를 제공하는 관점의 균형이 맞물려 있다. 애자일 팀은 출시 계획을 수립하는 과정에서 필요한 산출물의 종류와 문서 작성 수준을 결정해야 한다.

애자일 프로젝트의 일반적인 비즈니스 중심 산출물에는 사용자 스토리와 인수 기준이 포함된다. 사용자 스토리는 요구사항 명세의 애자일 양식으로, 시스템이 하나의 일관된 특성 또는 기능으로 어떻게 동작하는지에 대해 설명이 들어있어야 한다. 사용자 스토리는 하나의 반복주기에서 완료될 수 있도록 작은 기능으로 정의해야 한다.

관련된 기능의 큰 집합 혹은 하나의 복잡한 기능으로 조합할 수 있는 하위 기능의 집합을 “에픽”이라고 부른다. 에픽은 다른 개발팀을 위한 사용자 스토리를 포함할 수 있다. 예를 들어, 하나의 사용자 스토리에서 API 레벨(미들웨어)에서 필요한 사항에 대해 설명하면서 다른 스토리에서는 UI 레벨(애플리케이션)에서 필요한 사항에 대해 설명할 수 있다. 이러한 집합은 스프린트 전체에 걸쳐 개발될 수 있다. 각각의 에픽과 사용자 스토리는 연관된 인수 기준을 가져야 한다.

애자일 프로젝트의 일반적인 개발자 산출물에는 코드가 포함된다. 애자일 개발자 역시 종종 자동화된 단위 테스트를 만들 수 있다. 이러한 테스트는 코드 개발 이후에 작성되기도 한다. 그러나 개발자가 코드의 각 부분을 작성한 후 제품이 예상대로 동작하는지 여부에 대한 검증 방법을 제공하기 위해 코드의 각 부분이 기록되기 전에 순차적인 테스트를 작성하는 경우도 있다. 이러한 방법을 테스트 우선 혹은 테스트 주도 개발이라고 일컫지만, 실제 이러한 환경에서 작성하는 테스트 코드는 테스트라기 보다 실행 가능한 하위 레벨의 설계 명세에 가깝다.

애자일 프로젝트의 일반적인 테스터 작업 산출물에는 자동화된 테스트와 여러 문서들이 포함된다. 이 문서에는 테스트 계획, 품질 리스크 카탈로그, 수동 테스트 케이스, 결함 리포트, 결함 결과 로그들이 포함된다. 이러한 문서들은 가능한 간단히 작성한다(전통적인 개발 라이프사이클에서도 이러한 문서들은 일반적으로 간단히 작성된다). 또한 테스터는 결함 리포트와 테스트 결과 로그로부터 테스트 통계를 도출하며, 통계 역시 가능한 가벼운 방식으로 접근한다.

일부 특별한 경우, 가령, 규제, 안전 우선, 분산된 환경 혹은 매우 복잡한 프로젝트나 제품을 구현하는 경우, 일부 애자일 팀은 이러한 작업 산출물을 보다 공식적으로 작성해야 할 수 있다. 예를 들어, 일부 팀은 사용자 스토리와 인수 조건을 좀 더 공식화된 요구사항 명세의 형태로 기술해야 할 수도 있다. 2 차원 매트릭스(예: 수직 및 수평 추적) 보고서를 작성해서 감사, 규정 및 기타 요구 사항 만족여부를 확인해야 할 수 있다.

2.1.3 테스트 레벨

테스트 레벨은 대개 테스트 대상의 성숙도와 완성도에 관련된 테스트 활동들을 논리적으로 묶은 것이다.

순차 주기 모델에서는 테스트 레벨이 종종 어느 레벨의 종료 기준이 다음 레벨의 시작 조건의 일부라는 식으로 정의되기도 하지만, 일부 반복적 모델에서는 이 규칙이 적용되지 않는다. 테스트 레벨은 오버랩(중첩 혹은 동시 발생)된다. 요구 명세, 설계 명세 및 개발 활동은 테스트 레벨과 오버랩 될 수 있다.

일부 애자일 라이프사이클에서는 요구사항, 설계, 코드에 대한 변화가 반복주기 상의 모든 지점에서 발생할 수 있기 때문에 오버랩(중첩 또는 동시발생)이 발생한다. 스크럼에서는 이론적으로 반복주기 계획이 완료된 이후의 사용자 스토리가 변경되는 것을 허용하지 않으나, 실제 상황에서는 때로 변경이 발생한다. 반복주기가 진행되는 동안, 일반적인 경우 모든 사용자 스토리를 대상으로 다음과 같은 테스팅 활동을 순차적으로 적용한다:

- 단위 테스팅: 일반적으로 개발자가 수행한다

- 기능 인수 테스팅: 두 가지 활동으로 구분되어 수행되기도 한다
- 기능 검증 테스팅: 주로 자동화되어 개발자나 테스터에 의해 수행되며, 유저 스토리에 대한 인수 기준 검증을 포함한다
- 기능 확인 테스팅: 보통 수동으로 개발자, 테스터, 비즈니스 이해관계자가 함께 진행하며 구현된 기능이 사용 목적에 적합한지 확인하기 위해 개발자, 테스터 그리고 비즈니스 이해관계자의 협업이 포함될 수 있다

이와 함께 반복주기가 진행되는 동안 리크레션 테스팅이 병렬로 진행되는 경우가 있다. 이 병행 테스트 과정에는 현재 반복주기와 이전 반복주기에서의 자동화된 단위 테스트와 기능 베리피케이션 테스트가 수행되며, 지속적인 통합 프레임워크가 리그레션 테스트를 포함한다.

일부 애자일 프로젝트에서는 시스템 테스트 레벨을 적용하며, 해당 테스트 대상 사용자 스토리가 준비되는 즉시 시작된다. 시스템 테스트에는 기능 테스트는 물론 성능, 신뢰성, 사용성 및 다른 관련된 테스트 유형이 포함되기도 한다.

애자일 팀은 다양한 형태의 인수 테스트를 적용할 수 있다(ISTQB Foundation 실라버스 [ISTQB_FL_SYL]에서 설명한 용어 사용). 각 반복주기 마다, 각 반복주기가 종료된 이후 또는 일련의 반복주기가 종료된 이후 내부 알파 테스트, 외부 베타 테스트, 사용자 인수 테스트, 운영 인수 테스트, 규제 승인 테스트 및 계약 승인 테스트를 수행할 수 있다.

2.1.4 테스팅과 형상 관리

애자일 프로젝트에서는 무거운 자동화 도구들을 많이 사용해 소프트웨어 개발, 테스트 및 관리를 수행한다. 개발자들은 도구를 사용해 정적 분석, 단위 테스팅, 코드 커버리지 측정을 수행한다. 개발자들은 지속적으로 코드와 단위 테스트 코드를 형상 관리 시스템에 입력하며, 자동화된 빌드 및 테스팅 프레임워크를 사용한다. 이 프레임워크는 기존 시스템과 새로운 소프트웨어를 지속적으로 통합하도록 해주며, 통합 과정에서 새롭게 체크인 된 코드에 대한 정적 분석과 단위 테스트를 자동으로 수행한다 [Kubaczkowski].

자동화 된 테스트는 통합 및 시스템 레벨에서 수행되는 기능 테스트를 포함할 수 있다. 이러한 기능 자동화 테스트 케이스는 테스팅 하네스, 오픈 소스 사용자 인터페이스 기능 테스트 도구, 혹은 상용 도구들을 사용해 생성할 수 있으며, 지속적인 통합 프레임워크의 한 부분으로 실행되는 자동화 테스트와 통합되기도 한다. 기능 테스트에 시간이 오래 소요되는 경우에는 단위 테스트와 별도로 실행하기도

하는데, 예를 들어 새로운 소프트웨어(코드)가 체크인 될 때마다 단위 테스트를 실행하는 반면 기능 테스트는 며칠을 주기로 실행할 수도 있다.

자동화 테스트의 목적 중 하나는 빌드가 작동 가능하고 설치 가능한지를 확인하는 것이다. 자동화 테스트가 실패하는 경우 팀은 다음 코드를 체크인 하기 전에 발견된 결함들을 모두 수정해야 한다. 이러한 자동화 테스트 결과를 효과적으로 표시하기 위해 실시간 보고에 대한 투자가 필요하다. 이러한 접근법은 빌드가 깨지거나 소프트웨어 설치가 실패하는 원인을 빠르게 감지하기 때문에 많은 전통적인 프로젝트에서 발생할 수 있는 “빌드 -> 설치 -> 실패 -> 재빌드 -> 재설치” 과정에서 발생하는 비용의 낭비와 효율성 저하를 줄일 수 있다.

자동화 테스팅과 빌드 도구는 종종 애자일 프로젝트에서 발생하는 잦은 변경과 관련된 리그레션 리스크를 관리하는데 도움이 된다 [Jones11]. 그러나 단위 테스트의 제한된 결함 검출 효율성[Jones11] 때문에 이러한 리스크를 관리하기 위해 과도하게 자동화된 단위 테스트에만 의존하는 것은 문제가 될 수 있다. 통합 레벨과 시스템 레벨의 자동화 테스트도 요구된다.

2.1.5 독립적 테스팅을 위한 조직 구성 옵션

ISTQB Foundation 실라버스[ISTQB_FL_SYL]에서 설명하고 있는 바와 같이, 개발팀과 별도로 독립적으로 구성된 팀의 테스터들은 결함을 더 효과적으로 발견할 수 있는 장점이 있다. 일부 애자일 팀에서는 개발자가 자동화 테스트 형식으로 여러 테스트를 생성한다. 각 팀에는 한 명 혹은 그 이상의 테스터가 포함될 수 있으며 이들은 다양한 테스팅 작업을 수행한다. 그러나 개발팀 내에 테스터가 포함되는 경우 독립성과 객관적인 평가를 하지 못하게 되는 위험 요소가 존재하게 된다.

이와 달리 어떤 애자일 팀은 완전히 독립적이고 개별적인 테스터 팀을 유지하며 각 스프린트의 마지막 날에 필요에 따라 테스터를 할당한다. 이는 독립성을 보장할 수 있으며, 이러한 테스터들은 소프트웨어의 공정하고 객관적인 평가를 제공할 수 있다. 그러나 이러한 방법을 사용할 경우 시간이나 제품의 새로운 기능에 대한 이해가 부족할 수 있으며 비즈니스 이해관계자와 개발자 사이의 관계에서 문제가 발생할 여지도 있다.

세 번째 옵션은, 장기적인 관점에서는 독립되고 분리된 테스트 팀을 운영하고 프로젝트가 시작되는 시점에 각 테스터들을 애자일 팀에 할당하는 방법이다. 이 방법을 통해 테스트 팀은 독립성을 유지하면서도 제품에 대해 깊이 이해할 수 있으며 다른 팀 멤버들과도 강력한 관계를 유지할 수 있다. 또한 전문적인 역량을 가진 테스터들을 애자일 팀에 할당하지 않고, 이들이 보다 장기적이거나 반복주기와 별개로 수행하는 태스크 즉, 자동화 테스트 도구 개발, 비기능 테스팅 수행, 테스트 환경 및

데이터 구축 및 지원, 하나의 스프린트에서 완료할 수 없는 레벨 테스트 수행(예: 시스템 통합 테스트) 등을 수행하도록 할 수 있다.

2.2 애자일 프로젝트의 테스팅 상태

애자일 프로젝트에서는 빠르게 변경사항이 발생하며, 이는 테스트 상태, 테스트 진행 및 제품의 품질이 끊임없이 진화한다는 것을 의미한다. 테스터는 이와 관련된 정보들을 애자일 팀에 전달해서 팀이 각 반복주기의 성공적 완료를 위해 적절한 의사 전달을 할 수 있도록 지원해야 한다. 또한 변화는 이전 반복주기에서 기존의 기능들에 영향을 줄 수 있다. 따라서 수동 및 자동화 테스트를 지속적으로 업데이트하여 리그레션 리스크를 효과적으로 처리해야 한다.

2.2.1 테스트 상태, 진행 및 제품 품질 커뮤니케이션

애자일 팀은 각 반복주기의 마무리 지점에 적절히 동작하는 소프트웨어를 제공하며 프로세스를 진행한다. 동작하는 소프트웨어의 제공 가능 시점을 결정하기 위해 개발팀은 반복주기와 출시의 모든 아이템 진행 상황을 예의 주시해야 한다. 애자일 팀의 테스터들은 다양한 방법을 활용해 테스트 진척과 상태를 기록한다. 여기에는 테스트 자동화 결과, 애자일 태스크 보드 상의 테스트 태스크와 스토리 진행 내용, 팀의 진척을 보여주는 번다운 차트 등이 포함된다. 기록한 테스트 진척과 상태는 팀 내 다른 멤버들과 위키 대시보드나 대시보드 스타일의 이메일 혹은 스탠드업 미팅 시에 구두로 공유한다. 애자일 팀은 위키 스타일 대시보드와 이메일을 업데이트하여 작업 진행과 테스트 결과에 기반한 상태 보고서를 자동으로 생성하는 도구를 활용할 수 있다. 이러한 의사소통 수단을 통해 프로세스 개선에 사용할 수 있는 테스트 프로세스에서 메트릭을 수집할 수 있다. 또한 자동화된 방식으로 테스트 상태를 의사 소통하는 것은 더 많은 테스트 케이스를 설계하고 실행하는데 집중할 수 있도록 테스터의 시간을 절약해줄 수 있다.

팀은 번다운 차트를 활용해서 전체 출시 기간과 각 반복주기 기간 중의 진척상황을 추적하기도 한다. 번다운 차트[Crispin08]는 출시 또는 반복주기에 할당 된 시간 및 그 안에 수행할 남아있는 작업의 양을 나타낸다.

테스트 상태를 포함해 전체 팀의 현재 상태를 즉각적이고도 자세하게 시각화하기 위해 애자일 태스크 보드를 사용할 수 있다. 스토리 카드, 개발 태스크, 테스트 태스크, 반복주기 계획 중 만들어진 다른 태스크들(섹션 1.2.5 참조)은 태스크 보드에 표시하며, 각각 다른 색상의 카드를 사용해 태스크 종류를 구분한다. 반복주기가 진행되는 동안 이 태스크들의 진행 상황은 해당 태스크의 카드를 To do, WIP, Verify, Done 와

같이 태스크 보드 위의 열 위에서 이동하면서 관리된다. 애자일 팀은 대시보드나 상태 업데이트를 자동화하는 도구를 사용해서 스토리 카드와 애자일 태스크 보드를 관리하기도 한다.

작업 보드의 태스크 작업은 사용자 스토리에 대해 정의된 인수 기준에 관한 것이다. 테스트 작업에 대한 테스트 자동화 스크립트, 수동 테스트 그리고 탐색적 테스트가 완료 기준을 달성하면 작업은 작업 보드의 done 열로 이동한다. 전체 팀은 정기적(주로 일일 스탠드업 미팅에서)으로 태스크 보드의 상태를 리뷰함으로써 적절한 속도로 태스크들이 진행되고 있는지 확인한다. 전혀 진행되지 않거나 계획보다 너무 느리게 진행되는 태스크가 있는 경우, 팀은 해당 태스크를 리뷰하고 태스크의 진행을 방해하는 이슈들이 없는지 확인한다.

일일 스탠드업 회의에는 테스터를 포함해 애자일 팀의 모든 구성원이 참여한다. 이 회의는 각 구성원의 현재 상태 공유를 목적으로 하며 각 멤버는 다음 사항들을 공유한다 [Agile Alliance Guide]:

- 지난 일일 미팅 이후 완료한 작업은 무엇인가?
- 다음 회의까지 완료할 작업은 무엇인가?
- 현재 하고 있는 작업은 무엇인가?

테스트 진행을 방해할 수 있는 모든 문제도 일일 스탠드업 회의에서 다루어지므로, 전체 팀은 해당 문제를 함께 인식하고 적절하게 해결할 수 있다.

전반적인 제품의 품질을 향상시키기 위해 많은 애자일 팀은 고객 만족도 조사를 통해 제품이 고객의 기대를 충족시키는지에 대한 피드백을 수집한다. 팀은 테스트 성공/실패율, 결함 발견율, 확인과 리그레션 테스트 결과, 결함 밀도, 결함 발견과 수정률, 요구사항 커버리지, 리스크 커버리지, 코드 커버리지 그리고 제품 품질을 향상하기 위한 코드 변동과 같은 기존 개발 방법론에서 다루던 것과 유사한 다른 기준을 사용할 수 있다. 다른 라이프사이클과 동일하게 매트릭의 수집과 보고는 의사결정을 할 수 있도록 적절한 연관관계를 가지고 이루어져야 한다. 메트릭은 어떤 경우에도 구성원에 대한 보상, 처벌 또는 구분(차별)을 목적으로 사용되어서는 안 된다.

2.2.2 수동 및 자동화된 테스트 케이스 갱신을 통한 리스레션 리스크 관리하기

애자일 프로젝트에서는 각 반복주기가 종료됨으로써 제품이 완성되어 가며, 이와 함께 테스팅 범위 역시 증가한다. 현재 반복주기에서 발생한 코드 변경 테스팅과 함께, 테스터는 이전 반복주기에서 개발되고 테스트된 기능에 리그레션이 발생하지 않았는지 확인해야 한다. 애자일 개발에서 발생하는 리그레션 리스크로 인해 코드가 광범위하게 변경될 확률이 높다(현재 버전에서 다른 버전으로 코드 라인이 추가, 수정, 삭제됨). 변경에 대응하는 것이 애자일 원칙의 핵심이기 때문에, 비즈니스 요구를 만족시키기 위해

이미 고객에게 전달된 기능에도 변경 사항이 발생할 수 있다. 많은 기술 부채를 발생시키지 않으면서 개발 속도를 유지하기 위해, 팀은 가능한 조기에 모든 테스트 레벨에서 테스트 자동화에 투자해야 한다. 또한 모든 테스트 자산들, 예를 들어 자동화 테스트 케이스, 수동 테스트 케이스, 테스트 데이터 및 다른 테스팅 산출물 등은 매 반복주기가 수행될 때마다 최신 상태로 유지되어야 한다. 모든 테스트 자산들은 형상 관리 도구를 통해 관리하는 것이 바람직하다. 이를 통해 버전을 컨트롤하고 모든 팀 멤버가 해당 자산에 쉽게 접근하게 할 수 있으며 기존 테스트 자산의 이력 정보를 유지하면서도 기능 변경으로 인한 변화 내용을 적절하게 반영할 수 있다.

시간 압박이 강한 애자일 프로젝트의 경우 모든 테스트를 완전하게 반복하는 것이 거의 불가능하다. 따라서 테스터는 매 반복주기 마다 별도의 시간을 할당해 이전 반복주기와 현재 반복주기에서 수행한 수동 테스트 케이스와 자동화 테스트 케이스를 리뷰해서 리그레션 테스트 스위트에 할당 가능한 테스트 케이스를 선택해야 하고, 더 이상 수행할 필요가 없는 테스트 케이스는 제거한다. 초반의 반복주기에서 특정한 기능을 테스트하기 위해 작성한 테스트 케이스는 반복주기 후반에서는 그 가치가 작아질 수 있는데, 이는 기능 자체가 변경되거나 새로운 기능이 해당 기능의 동작 방식을 변경할 수 있기 때문이다.

테스트 케이스를 검토하는 동안, 테스터는 자동화에 대한 적합성을 고려해야 한다. 팀은 이전과 현재 반복주기에서 가능한 많은 테스트를 자동화 할 필요가 있다. 이를 통해 자동화된 리그레션 테스트를 수행함으로써 리그레션 테스트를 수동으로 수행했을 때보다 적은 공수를 사용해 리그레션 리스크를 줄일 수 있고, 테스터는 남은 시간을 현재 반복주기의 새로운 기능을 더 철저하게 테스트하는데 사용할 수 있다.

테스터는 현재 반복주기에서 발생한 변경에 의해 영향을 받은 이전 반복주기 및 출시 테스트 케이스를 확인하고 신속하게 업데이트 할 수 있는 역량을 반드시 가지고 있어야 한다. 팀이 테스트 케이스를 어떻게 설계하고 작성하고 저장할 지의 결정은 출시 계획 동안 수립한다. 테스트 설계 및 구현과 관련된 좋은 실천방법을 조기에 채택해 일관적으로 적용해야 한다. 각 반복주기에 대한 짧은 테스팅 일정과 지속적으로 발생하는 변경은 부족한 테스트 설계와 구현 실천방법의 영향을 증가시킬 것이다.

자동화 테스트를 적용함으로써 애자일 팀은 모든 테스트 레벨에서 제품 품질 관련 피드백을 빠르게 얻을 수 있다. 잘 작성된 자동화 테스트는 현재 시스템의 기능을 생생하게 기술한다 [Crispin08]. 자동화 된 테스트 케이스와 해당 테스트 케이스의 결과를 해당 제품의 빌드 버전과 함께 형상 관리 시스템에 저장함으로써 애자일 팀은 언제든 테스트 기능과 그 결과를 확인할 수 있게 된다.

자동화된 단위 테스트 코드에서 소스코드를 형상 관리 시스템의 메인 라인에 체크인하기 전에 수행하여, 변경한 코드가 소프트웨어 빌드를 중단시키지 않도록 해야 한다. 빌드가 깨지면 전체 팀의 업무 속도가 떨어지며, 이를 방지하기 위해서 자동화된 단위 테스트를 모두 통과한 코드만을 메인 라인에 체크인해야 한다. 자동화된 단위 테스트 결과는 코드와 빌드 품질에 대한 즉각적인 피드백을 제공하기는 하지만, 제품 품질의 좋고 나쁨에 대한 피드백을 제공하지는 못한다.

지속적인 통합을 전체 시스템 빌드에 적용하는 경우, 자동화된 인수 테스트를 정기적으로 실행한다. 전체 시스템을 대상으로 최소한 1 회 이상 실행하며, 일반적으로 개별적인 코드 체크인이 발생하는 경우에는 실행하지 않는다. 이러한 테스트는 자동화된 단위 테스트를 수행할 때 보다 많은 시간을 필요로 하며 결과적으로 코드 체크인 속도를 저하시킬 수 있기 때문이다. 자동화 된 인수 테스트 시험 결과는 마지막 빌드의 리그레션에 대한 제품 품질 피드백은 제공하지만 전반적인 제품 품질 상태를 제공하지는 않는다.

자동화 테스트는 전체 시스템을 대상으로 지속적으로 실행할 수 있다. 핵심적인 시스템 기능과 통합 지점을 검증하기 위한 자동화 테스트의 초기 집합은 새로운 빌드가 테스트 환경에 배포되는 즉시 생성되어야 한다. 이러한 테스트는 일반적으로 빌드 베리피케이션 테스트로 알려져 있다. 빌드 베리피케이션 테스트 결과는 배포된 소프트웨어에 대한 즉각적인 피드백을 제공하므로 팀은 불안정한 빌드를 테스트하기 위해 시간을 소모하지 않아도 된다.

리그레션 테스트 셋에 포함된 자동화 테스트는 지속적 통합 환경에서 매일 주요 빌드의 일부로 실행되고, 테스트 환경에 새로운 빌드가 배포될 때 다시 실행된다. 자동화된 리그레션 테스트가 실패하는 즉시, 팀은 다른 작업을 멈추고 테스트가 실패한 이유를 조사한다. 현재 반복주기에서 정상적인 절차에 의해 변경된 기능으로 인해 테스트가 실패할 수 있으며, 이러한 경우 테스트 케이스 및 사용자 스토리는 새로운 인수 기준에 맞게 업데이트 되어야 한다. 이와 반대로 다른 테스트가 이미 변경된 내용을 커버하는 경우, 해당 테스트는 더 이상 필요하지 않을 수도 있다. 어떤 경우든 결함으로 인해 테스트가 실패하게 되면, 새로운 기능을 추가하기 전에 결함을 조치할 수 있으므로 팀에게 도움이 된다.

테스트 자동화와 더불어 다음의 테스트 업무들도 자동화가 가능하다:

- 테스트 데이터 생성
- 시스템의 테스트 데이터 기록
- 테스트 환경에 빌드 배포
- 테스트 기준점으로 테스트 환경(데이터 베이스 혹은 웹사이트 데이터 파일 등) 복원
- 데이터 결과 비교

이러한 태스크들을 자동화함으로써 해당 태스크를 수행하는데 필요한 오버헤드를 줄일 수 있으며, 팀은 개발과 새로운 기능 테스트에 집중할 수 있다.

2.3 애자일 팀의 테스터 역할과 역량

애자일 팀에서 테스터는 반드시 모든 다른 팀 구성원 및 비즈니스 이해관계자들과 협력해야만 한다. 이는 테스터가 가지고 있어야 할 역량과 애자일 팀에서 테스터들이 수행해야 하는 활동과 관련하여 많은 시사점을 제공한다.

2.3.1 애자일 테스터의 역량

애자일 테스터는 ISTQB Foundation Level 실라버스에서 언급된 모든 역량을 보유하고 있어야 한다. 이뿐 아니라 테스트 자동화, 테스트 주도 개발, 인수 테스트 주도 개발, 화이트박스, 블랙박스 테스팅은 물론 경험기반 테스팅 역량도 보유해야 한다.

애자일 방법론은 협업, 팀 멤버 및 팀 외부 이해관계자화의 상호 작용을 매우 중시하기 때문에 애자일 팀의 테스터는 다음과 같은 뛰어난 대인 관계 역량을 가지고 있어야 한다:

- 팀 멤버들 및 이해관계자와 협업 시 긍정적이며 문제 해결 중심적인 태도로 임한다
- 제품과 관련해 핵심적이고, 품질 중심적이며, 비판적인 생각을 제안한다
- 작성된 명세에 전적으로 의존하기보다는 이해관계자로부터 적극적으로 정보를 획득한다
- 테스트 결과, 테스트 진척 및 제품 품질 등에 대해 정확하게 평가하고 보고한다
- 고객 대표자 및 이해관계자들과의 효과적인 협업을 통해 테스트 가능한 사용자 스토리, 특히 인수 기준을 정의한다
- 프로그래머 및 다른 팀 멤버 등 팀 내에서 협업한다
- 변경, 추가 또는 테스트 개선 등 변화에 빠르게 대응한다
- 스스로 작업을 조직화하고 계획을 수립한다

애자일 팀에 소속된 테스터를 포함하여 모든 테스터는 대인 관계 역량을 지속적으로 키워나가야 한다.

2.3.2 애자일 팀의 테스터의 역할

애자일 팀의 테스터는 테스트 상태, 테스트 진척, 제품 품질에 대한 피드백을 생성하고 제공하는 것뿐만 아니라 프로세스 품질에 대한 피드백도 함께 제공해야 한다. 이 실라버스에서 기술된 활동들뿐만 아니라 다음과 같은 활동도 수행해야 한다:

- 테스트 전략을 이해하고 실행하며 업데이트 한다
- 적용 가능한 모든 커버리지 영역에서 테스트 커버리지를 측정하고 보고한다
- 적절한 테스팅 도구의 사용을 보장한다
- 테스트 환경과 테스트 데이터를 구성하고 활용하며 관리한다
- 결함을 보고하고 해당 결함을 해결하게 위해 팀과 협업한다
- 팀 멤버들에게 테스팅 관련 코칭을 제공한다
- 출시 계획 및 반복주기 계획에 적절한 테스팅 업무를 반영한다
- 개발자 및 비즈니스 이해관계자들과 능동적으로 협력해 요구사항, 특히 테스트 가능성, 일관성, 완전성 측면을 명확화 한다
- 팀 회고에 적극적으로 참여하여 개선안을 제안하고 구현한다

애자일 팀의 모든 멤버는 제품 품질에 대해 책임을 지며 테스트와 관련된 태스크들을 수행한다.

애자일 조직은 다음과 같은 테스트와 관련된 조직적인 리스크를 마주할 수 있다:

- 테스터가 개발자와 너무 밀접하게 작업하여 적절한 테스터 사고 방식을 잃을 수 있다
- 테스터가 팀 내에서 비효율적, 비효과적 또는 낮은 품질 관행에 대해 관용 또는 침묵할 수 있다
- 테스터가 제한된 시간의 반복주기에서 발생하는 변화의 속도를 따라갈 수 없다

이러한 리스크를 완화하기 위해 2.1.5 절에서 언급했듯이 테스터의 독립성을 보장할 수 있는 조직 형태를 고려해야 할 수 있다.

3. 애자일 테스팅 방법, 기법 및 도구 – 480 분

키워드

인수 조건, 탐색적 테스팅, 성능 테스팅, 제품 리스크, 품질 리스크, 회귀 테스팅, 테스트 접근법, 테스트 차터, 테스트 추정, 테스트 자동화, 테스트 전략, 테스트 주도 개발, 단위 테스트 프레임워크

학습 목표

3.1 애자일 테스팅 방법

FA-3.1.1 (K1) 테스트 주도 개발, 인수테스트 주도 개발, 행위 주도 개발의 개념을 상기한다

FA-3.1.2 (K1) 테스트 피라미드의 개념을 상기한다

FA-3.1.3 (K2) 테스팅 사분면과 각 사분면의 테스팅 레벨 및 테스팅 종류와의 상관관계를 설명한다

FA-3.1.4 (K3) 애자일 프로젝트의 스크럼 팀에서 테스터 역할을 수행할 있다

3.2 품질 리스크 식별 및 테스트 노력 추정

FA-3.2.1 (K3) 애자일 프로젝트의 품질 리스크를 식별한다

FA-3.2.2 (K3) 품질 리스크와 반복주기 내용에 기반하여 테스팅에 필요한 자원을 추정한다

3.3 애자일 프로젝트의 기법들

FA-3.3.1 (K3) 테스팅 활동을 지원하기 위한 관련 정보를 이해한다

FA-3.3.2 (K2) 비즈니스 이해관계자들이 테스트 용이성이 있는 인수 조건을 정의하도록 설명한다

FA-3.3.3 (K3) 주어진 유저스토리에 대해 인수 테스트 주도 개발의 테스트케이스를 작성한다

FA-3.3.4 (K3) 주어진 유저스토리에 대해 블랙박스 테스트 설계 기법을 활용하여 기능적, 비기능적 행위의 테스트 케이스를 작성한다

FA-3.3.5 (K3) 탐색적 테스팅을 수행하여 애자일 프로젝트의 테스팅을 지원한다

3.4 애자일 프로젝트의 도구들

FA-3.4.1 (K1) 애자일 프로젝트에서의 테스팅 목적과 활동에 맞춰 적용 가능한 다양한 도구들을 상기한다

3.1 애자일 테스팅 방법

애자일 적용 여부와 관계없이 높은 품질의 제품을 만들고자 하는 모든 개발 프로젝트에 적용 가능한 테스팅 실천방법들이 있다. 이 실천방법들은 적절한 동작을 표현하기 위해 미리 테스트를 작성하고, 초기 결함의 예방, 발견 및 제거에 초점을 맞추고, 적절한 테스트가 적절한 시간에 올바른 테스트 레벨의 활동으로 수행되는 것을 보장한다. 이런 실천방법을 널리 전파하고자 하는 것이 애자일 실천가들의 목표이다. 애자일 프로젝트에서 테스터는 전체 개발 수명주기에 걸쳐 테스팅 방법을 안내하는 핵심적인 역할을 하게 된다.

3.1.1 테스트 주도 개발, 인수 테스트 주도 개발, 행위 주도 개발

테스트 주도 개발, 인수 테스트 주도 개발, 행위 주도 개발은 다양한 테스트 레벨의 테스트를 수행하기 위해 애자일 팀이 사용하는 세 가지 상호 보완적인 방법이다. 코드 작성 전에 테스트가 정의되어 있기 때문에 각각의 기술은 개발 초기의 테스트 및 QA 활동이 주는 효과에 대한 좋은 사례가 된다.

테스트 주도 개발

테스트 주도 개발 (TDD)는 자동화된 테스트 케이스에 맞추어 코드를 개발하는 방법이다. 테스트 주도 개발을 위한 프로세스는 다음과 같다:

- 테스트(코드)를 추가한다. 테스트 코드는 구현되어야 할 기능에 대한 프로그래머의 개념을 확인하기 위한 작은 코드이다;
- 테스트를 실행한다. 코드가 존재하지 않기 때문에 테스트는 실패한다;
- 코드를 작성하고 테스트가 통과할 때까지 계속 테스트를 실행하고 이를 반복한다;
- 테스트가 통과되면 코드를 리팩토링한다. 리팩토링 이후의 정상 동작여부를 파악하기 위해 테스트를 실행한다;
- 코드의 다음 조각에 대해서 이 과정을 반복한다.

테스트는 주로 코드에 집중된 단위테스트 레벨 위주지만, 통합 또는 시스템 테스트 레벨에서도 진행될 수 있다. 테스트 주도 개발은 익스트림 프로그래밍을 통해 인기를 얻었지만, 다른 애자일 방법론이나 폭포수 방법론에서도 사용 가능하다. 테스트 주도개발은 개발자가 명확하게 정의된 예상 결과에 초점을 맞출 수 있도록 해준다. 작성된 테스트는 테스트 자동화와 지속적인 통합에 사용된다.

.

인수 테스트 주도 개발

인수 테스트 주도 개발은 사용자 스토리를 작성하는 동안 인수 기준 및 테스트 방법을 정의한다 (1.2.2 절 참조). 인수 테스트 주도 개발은 모든 이해 관계자가 소프트웨어 구성요소가 동작하는 방식을 이해하고,

개발자, 테스터 및 업무 대표자가 이 동작을 보장하기 위해 필요한 테스트를 함께 만드는 공동 접근 방식이다. 인수 테스트 주도 개발의 과정은 3.3.2 절에 설명되어 있다.

인수 테스트 주도 개발은 회귀 테스트를 위한 재사용 가능한 테스트를 만든다. 지속적인 통합 프로세스 내부에서 이러한 테스트의 생성 및 실행을 지원하는 도구들도 있다. 이런 도구는 시스템 테스트 또는 인수 테스트 레벨에서 실행할 수 있도록 프로그램의 데이터 및 서비스 층을 연결할 수도 있다. 인수 테스트 주도 개발은 결함과 기능 동작의 검증을 신속하게 해결할 수 있다. 이때 인수 기준은 기능에 대한 완료 여부를 결정하는 데 도움이 된다.

행위 주도 개발

행위 주도 개발 [Chelimsky10]에서는 개발자가 기대하는 소프트웨어 동작에 기반하여 코드를 테스트하는데 초점을 맞출 수 있다. 테스트가 소프트웨어의 특정 행위를 기반으로 하기 때문에, 일반적으로 다른 팀 구성원 및 이해 관계자가 이해하기 쉽다.

특정 행위 주도 개발 프레임워크는 Given/When/Then 의 형식으로 인수 기준을 정의할 수 있다:

Given 특정 상황에서,
When 이벤트가 발생하면,
Then 다음 몇 가지 결과를 확인한다

행위 주도 개발 프레임워크는 요구사항으로부터 도출된 테스트 케이스를 개발자가 사용할 수 있는 코드로 변환한다.

행위 주도 개발은 개발자가 비즈니스 요구 사항에 초점을 맞추어 정확한 단위 테스트를 정의하여 테스터 등 다른 이해 관계자들과 협력하는 것을 돕는다.

3.1.2 테스트 피라미드

소프트웨어 시스템은 다양한 레벨에서 테스트 될 수 있다. 일반적인 테스트 레벨([ISTQB_FL_SYL] 2.2 절 참조)은 피라미드의 맨 아래부터 위로 순차적으로 단위, 통합, 시스템 및 인수 테스트 순으로 존재한다. 피라미드는 하위 레벨(피라미드의 바닥)에서는 테스트의 양이 강조되고, 상위 레벨(피라미드의 상위)로 올라갈수록 테스트의 양은 감소한다. 보통 단위 및 통합 레벨의 테스트는 자동화 및 API 기반 도구를 사용하여 생성된다. 시스템 및 인수 레벨에서 자동화된 테스트는 GUI 기반 도구를 사용하여 생성된다. 테스트 피라미드 개념은 조기 결함 검출의 원리(즉, 가능한 한 빨리 결함을 찾아서 제거)에 기초한다.

3.1.3 테스팅 사분면, 테스트 레벨, 테스트 유형

브라이언 머릭이 정의한 테스팅 사분면은 각 테스팅 레벨을 애자일 방법론에서 사용하는 테스팅 유형으로 잘 분류하고 있다. 테스트 사분면 모델 및 그 변종은 모든 중요한 테스트 유형과 테스트 레벨이 개발 수명 주기에 포함되어 있는지 확인하는데 도움이 된다. 또한, 이 모델은 개발자, 테스터, 업무 대표자 등 모든 이해관계자에게 테스트의 유형을 구별하고 설명하는 방식이 된다.

테스트 사분면에서 테스트는 비즈니스에 직면(사용자)하거나 또는 기술(개발자)에 직면한 것으로 분류될 수 있다.
일부 테스트는 애자일 팀에 의해 수행된 작업을 지원하고, 소프트웨어의 동작을 확인한다. 다른 테스트는 제품을 확인할 수도 있다. 테스트는 완전 수동 혹은 완전 자동 혹은 자동과 수동의 조합으로 만들어질 수 있다. 네 개의 사분면은 다음과 같다:

- 1 사분면은 단위 레벨, 기술적 측면이며, 개발자를 지원한다. 여기에는 단위 테스트가 포함된다. 1 사분면의 테스트는 자동화되어야 하고, 지속적인 통합 절차에 포함되어야 한다.
- 2 사분면은 시스템 레벨에서 비즈니스에 직면하고 있다. 여기에서는 제품의 동작을 확인한다. 따라서, 기능 테스트, 스토리 테스트, 사용자 경험 프로토타입 및 시뮬레이션이 포함되어 있다. 2 사분면의 테스트는 인수 기준을 확인하며, 수동 또는 자동으로 수행된다. 인수 기준은 종종 사용자 스토리를 개발하는 동안 생성되고, 이는 스토리의 품질을 향상시키는데 도움이 된다. 또한, 자동화된 회귀 테스트 스위트를 만들 때에도 유용하다.
- 3 사분면은 시스템 또는 제품 인수 테스트 레벨에서 비즈니스에 직면하고 있다. 여기에서는 사실적인 시나리오와 데이터를 사용하여 제품을 비판하는 테스트가 포함되어 있다. 3 사분면은 시나리오 및 프로세스 흐름 테스트, 사용성 테스트, 사용자 인수 테스트, 알파 테스트, 베타 테스트를 포함한다. 이런 테스트는 종종 수동이며 사용자 중심적이다.
- 4 사분면은 시스템 또는 운영 인수 테스트 레벨에서 기술에 직면하고 있다. 여기에서는 제품의 비판 테스트가 포함되어 있다. 4 사분면은 성능, 부하, 스트레스, 및 확장성 테스트, 보안 테스트, 유지 관리, 메모리 관리, 호환성 및 상호 운용성, 데이터 마이그레이션, 인프라 및 복구 테스트를 포함한다. 이런 테스트는 보통 자동화되어 있다.

반복주기를 반복하는 동안, 모든 사분면에 걸쳐 테스트를 진행해야 할 수도 있다. 테스트 사분면은 정적 테스트보다는 동적 테스트에 더 적합하다.

3.1.4 테스터의 역할

실라버스 전반에 걸쳐 애자일 방법론과 기법, 다양한 애자일 수명 주기 내에서 테스터의 역할을 참조했다. 이번 섹션은 스크럼 수명 주기[Aalst13]를 따르는 프로젝트에서 테스터의 역할에 대해 특별히 살펴볼 것이다.

팀워크

팀워크는 애자일의 기본 원칙이다. 애자일는 함께 일하는 개발자, 테스터 및 업무 대표자로 구성된 전체 팀 접근 방식을 강조한다. 모범적인 스크럼 팀의 특징은 다음과 같다:

- 교차 기능팀: 각 팀 구성원이 팀에 다양한 기술 영역을 제공한다. 이 팀은 테스트 전략, 테스트 계획, 테스트 설계, 테스트 실행, 테스트 평가 및 시험 결과 보고를 함께 진행한다.
- 자기 조직화: 팀은 개발자로만 구성할 수도 있지만, 2.1.5 절에서 언급 한 바와 같이 최소 한 명 이상의 테스터를 포함하는 것이 이상적이다.
- 공간 공유: 테스터는 개발자 및 제품 소유자와 같은 공간에서 함께 일한다.
- 협업: 테스터들은 자신의 팀 구성원, 다른 팀, 이해 관계자, 제품 소유자 및 스크럼 마스터와 공동 작업을 수행한다.
- 위임: 필요한 경우 설계와 테스트에 관한 기술 결정은 제품 소유자 및 다른 팀과 협력하여 팀 전체(개발자, 테스터, 그리고 스크럼 마스터)가 만든다.
- 헌신: 테스터는 질문과 기대와 고객과 사용자의 요구 사항과 관련하여 제품의 동작 및 특성을 평가하기 위해 최선을 다한다.
- 투명성: 개발 및 테스트의 진행 상황을 애자일 상황판에서 볼 수 있다 (2.2.1 절 참조).
- 신뢰: 테스터는 테스트 설계 및 실행 전략에 대한 신뢰를 확보해야 한다. 그렇지 않으면 이해 관계자가 테스트 결과를 신뢰하지 않는다. 이는 종종 테스트 프로세스에 대한 정보를 이해 관계자에게 제공함으로써 만들어지기도 한다.
- 열린 피드백: 피드백은 특히 애자일 프로젝트에서 성공의 중요한 기초이다. 회고는 팀이 성공과 실패에서 배우고 성장하도록 만든다.
- 탄력성: 테스팅는 애자일 프로젝트의 다른 모든 활동과 마찬가지로 변화에 기민하게 대응할 수 있어야 한다.

이러한 모범적인 팀의 특성은 스크럼 프로젝트에서 테스트의 성공 가능성을 극대화하는데 도움이 된다.

스프린트 제로

스프린트 제로는 프로젝트의 첫 번째 반복주기로 많은 준비 활동을 포함하고 있다 (1.2.5 참고). 테스터는 스프린트 제로에서 팀과 협업해 다음 활동을 수행 한다:

- 프로젝트의 범위 정의 (예: 제품 백로그)
- 초기 시스템 아키텍처와 프로토타입 생성
- 필요한 도구를 계획, 수급, 설치 (예: 테스트 관리, 결함 관리, 테스트 자동화, 형상 관리)
- 모든 테스트 레벨에 걸쳐 테스트 범위, 기술적 리스크, 테스트 종류(섹션 3.1.3 참고), 커버리지 목표를 포함한 초기 테스트 전략 수립
- 초기 품질 리스크 분석 수행 (섹션 3,2.1 참고)
- 테스트 프로세스, 진척율, 제품 품질을 측정하기 위한 품질 지표 정의
- "완료"의 의미 정의
- 태스크 보드 생성(섹션 2,2,1 참고)
- 고객에게 제품을 전달하기 전에 테스팅의 지속 및 중단 여부 정의

스프린트 제로에서는 전체 스프린트가 진행되는 동안 어떤 테스팅이 수행되어야 하는지, 어떻게 테스팅을 진행할지에 대한 방향을 설정한다.

통합

애자일 프로젝트에서 이상적인 목표는(가능한 모든 스프린트에서) 지속적으로 고객에게 가치를 전달하는 것이다. 이를 위해서 통합 전략은 설계와 테스트를 모두 고려해야 한다. 출시할 기능과 특징을 지속적으로 테스트하려면, 기본 기능과 기능 사이의 모든 종속 관계를 확인하는 것이 중요하다.

테스트 계획

테스트가 애자일 팀에 완전히 통합되면, 테스트 계획은 릴리즈 계획 시 시작되고 각 스프린트 동안 업데이트 된다. 각 릴리즈와 스프린트의 테스트 계획은 섹션 1.2.5 에서 제시한 내용을 다루어야 한다.

스프린트 계획의 결과, 모든 일은 하루나 이틀 분량의 태스크로 정의되어 태스크 보드에 기재된다. 또한, 모든 테스팅 이슈는 테스팅의 흐름을 지속적으로 유지하기 위해 추적되어야 한다.

애자일 테스팅 실천방법

스크럼 팀에서 테스터는 많은 실천 방법들을 적용할 수 있으며, 여기에는 다음과 같은 실천방법들도 포함된다:

- 페어링: 두명의 멤버(예: 개발자와 테스터, 테스터와 테스터, 테스터와 제품 소유자)가 테스트 또는 다른 작업을 수행하기 위해 하나의 동일한 작업 공간 앞에 함께 앉아 공동으로 태스크를 수행한다.
- 점진적인 테스트 설계: 사용자 스토리 및 기타 테스트 베이시스로부터 만들어진 테스트 케이스 및 차터가 점진적으로 간단한 테스트에서 시작해서 더 복잡한 테스트를 향해 발전한다.

- 마인드맵: [Crispin08] 테스트할 때 마인드맵은 유용한 도구이다. 예를 들어, 테스터는 테스트 전략을 표시하거나, 테스트 데이터를 설명하거나, 수행된 테스트 세션을 식별하기 위해 마인드맵을 사용할 수 있다.

위에 언급된 실천방법들을 포함한 다른 실천방법들은 ISTQB Foundation 실라버스 4 장에 설명되어 있다.

3.2 품질 리스크 식별 및 테스트 노력 추정

테스팅의 일반적인 목적은 릴리즈 이전에 허용 가능한 수준으로 품질 문제의 리스크를 감소시키는 것이고, 애자일이든 전통적인 프로젝트이든 이러한 목적은 동일하다. 그러나, 짧은 반복주기와 애자일 프로젝트의 잦은 변화에 대응하기 위해서 일부 변형이 필요하다.

3.2.1 애자일 프로젝트에서 품질 리스크 식별하기

테스팅에서 직면하는 도전 중 하나는 테스트 컨디션을 적절하게 선택하고, 할당하며 우선 순위를 선정하는 것이다. 이것은 각 시험 조건을 커버하기 위해 할당할 적절한 노력의 양을 결정하고, 테스트 작업의 효율성 및 효율을 최적화하기 위한 테스트 절차를 포함한다. 많은 제약 사항과 변수를 고려해야 하지만, 리스크 식별, 분석 및 리스크 완화 전략은 애자일 팀의 테스터가 실행할 테스트 케이스의 수를 결정하는 데 도움이 된다.

리스크란 부정적이거나 원하지 않는 결과 혹은 이벤트가 발생할 수 있는 가능성을 의미한다. 리스크의 수준은 발생 가능성과 영향도를 평가함으로써 파악할 수 있다. 제품의 품질에 잠재적인 문제가 있는 경우, 이 잠재적인 문제는 품질 리스크 또는 제품 리스크로 나타난다. 프로젝트에 잠재적인 문제가 있는 경우, 이 잠재적인 문제는 프로젝트의 리스크 또는 계획 과정의 리스크로 나타난다 [vanVeenendaal12] [Black07].

애자일 프로젝트에서 품질 리스크 분석은 두 군데에서 이루어진다:

- 릴리즈 계획: 기능의 리스크에 대해 전반적으로 파악하고 있는 업무 대표자와 테스터를 포함한 전체 팀이 리스크를 식별하고 평가한다.
- 반복주기 계획: 전체 팀이 품질 리스크를 식별하고 평가한다.

시스템 품질 리크스에는 다음 사항들이 포함된다:

- 보고서의 잘못된 계산식(기능적 리스크로 정확성과 관련됨)
- 사용자 입력에 대한 느린 응답(비기능 리스크로 효율성 및 응답 시간과 관련됨)
- 이해하기 어려운 화면 구성 및 입력 필드(비기능 리스크로 사용성 및 이해도와 관련됨)

앞서 언급한 바와 같이 반복주기는 반복주기 계획에서 시작하며, 반복주기 계획은 추정한 태스크들을 태스크 보드에 붙임으로써 마무리된다.
이 태스크들은 해당 태스크들과 관련된 품질 리스크의 수준에 기초해 우선 순위를 정할 수 있다.
높은 리스크과 관련된 작업은 빠르게 시작하고 더 많은 테스트 노력을 배정해야 하며, 낮은 리스크와 관련된 작업은 나중에 시작하고 테스트 노력을 상대적으로 덜 배정해야 한다.

다음은 반복주기 계획 시 애자일 프로젝트의 품질 리스크의 분석 과정을 예로 든 것이다:

1. 애자일 팀원 전체를 모은다. 물론 테스터도 포함된다;
2. 현재 반복주기의 모든 백로그 아이템을 나열한다;
3. 적절한 품질 특성을 고려하여 각 아이템에 대한 품질 리스크를 식별한다;
4. 식별한 리스크를 진단한다. 리스크를 분류하고 영향도와 발생 빈도에 따라 리스크의 수준을 파악한다;
5. 리스크의 수준에 따라 테스트의 수준을 결정한다;
6. 리스크, 리스크 수준, 품질 특성을 고려하여 각 리스크를 해결하기 위한 적절한 테스트 기법을 선택한다.

이후 테스터는 테스트 설계, 구현, 실행을 통해 리스크를 완화한다. 테스트는 고객, 사용자, 이해 관계자의 만족도에 영향을 미치는 기능, 행동, 품질 특성 및 속성 전체를 포함한다.

프로젝트 전반에 걸쳐 팀은 리스크 또는 알려진 품질 리스크 수준을 변화시키는 정보에 대해 항상 파악하고 있어야 한다. 테스트에 대한 결과를 반영하여 주기적으로 품질 리스크를 분석하고 조절한다. 조절 단계에는 기준 리스크 수준의 재평가, 새로운 리스크의 식별, 리스크 완화 활동에 대한 효과 평가 등이 포함된다.

테스트 실행 전에도 품질 리스크를 완화시킬 수 있다. 예를 들어, 사용자 스토리 관련 문제가 리스크 식별 과정에서 발견된 경우, 프로젝트 팀은 완화 전략으로 사용자 스토리를 철저하게 리뷰할 수 있다.

3.2.2 리스크와 내용을 근거로 테스트 노력 추정하기

출시 계획 과정에서 애자일 팀은 출시를 완료하기 위한 공수를 추정하며, 여기에는 물론 테스팅을 위한 공수도 포함된다. 애자일 프로젝트에서는 일반적으로 플래닝 포커 기법을 사용해 공수를 추정하는데, 이 기법은 참여자의 합의에 기반을 두고 있다. 제품 소유자 또는 고객은 공수를 추정해야 할 사용자 스토리를 읽는다. 스토리의 공수를 추정하는 인원들은 피보나치 수열(0, 1, 2, 3, 5, 8, 13, 21, 34, 55, 99, ……)과 같은 값 혹은 다른 방식의 순차적으로 증가하는 값(예: 가장 작은 것부터 가장 큰 셔츠의 크기)이 적힌 한 세트의 카드를 가지고 있다. 카드의 각 숫자는 해당 팀이 추정한 공수를 의미하며, 스토리의 크기에 따라

공수에 대한 불확실성이 기하급수적으로 증가할 수 있으므로, 피보나치 열을 사용할 것을 권장한다. 높은 추정치는 일반적으로 스토리가 잘 이해되지 않거나 여러 개의 작은 스토리로 분해되어야 한다는 것을 의미한다.

추정에 참여하는 사람들은 기능에 대해 논의하며 필요하다고 생각되는 모든 질문을 제품 소유자에게 던져야 한다. 추정을 함에 있어서 개발 및 테스팅 공수, 스토리의 복잡성, 테스팅 범위 등의 내용도 고려해야 한다. 따라서, 플래닝 포커가 시작되기 전에, 제품 소유자가 정한 우선 순위에 추가로 백로그 항목의 리스크 레벨을 표기하는 것이 바람직하다. 기능을 논의할 때, 각 추정자는 개인적으로 자신의 추정치를 대표하는 하나의 카드(숫자)를 선택한다. 모든 사람이 카드를 다 고르면 카드를 동시에 공개한다. 모든 추정치가 같은 카드일 경우 그 값을 추정치로 사용하고, 같은 추정치의 카드가 아닌 경우, 추정에 참여한 사람들은 추청치가 상이한 이유에 대해 논의하고 동일한 추정치에 도달할 때까지 추정을 반복한다. 포커 라운드 수를 제한하기 위해 카드 사용 규칙(예를 들면, 최종적으로 평균값을 사용하거나 최대값을 선택할 수 있다)을 별도로 정의할 수 있다.
이 토론은 제품 소유자가 요청한 제품 백로그 항목에서 해야 할 작업에 대한 집단의식을 향상시키고, 추정의 신뢰성을 높이는 데 도움이 된다 [Cohn04].

3.3 애자일 프로젝트의 테스트 기법들

전통적인 프로젝트에 적용된 테스트 및 테스트 기법의 대부분은 애자일 프로젝트에도 적용할 수 있다. 다만 테스트 기법, 용어 및 문서화와 관련해 조금 더 주의를 기울여야 할 차이점들이 있다.

3.3.1 인수 조건, 적절한 커버리지, 테스팅을 위한 기타 정보

애자일 프로젝트는 프로젝트의 초기 요구 사항을 백로그 우선 순위에 따라 사용자 스토리에 기재한다. 초기 요구 사항은 짧고 일반적으로 미리 정의된 형식을(1.2.2 절 참조) 따른다. 사용성 및 성능과 같은 비기능 요구 사항 역시 똑같이 중요하며, 이들은 고유의 사용자 스토리 혹은 다른 기능 사용자 스토리에 연결하여 명세를 작성할 수 있다. 비기능 요구 사항은 미리 정의된 형식이나 ISO25000 과 같은 국제 표준 또는 특정 산업 표준을 따를 수 있다.

사용자 스토리는 중요한 테스트 베이시스의 하나이며, 이 외에 다음과 같은 요소들을 테스트 베이시스에 포함시킬 수 있다:

- 이전 프로젝트의 경험

- 해당 시스템에 이미 존재하는 기능 및 품질 특성
- 코드, 아키텍처, 디자인
- 사용자 프로파일 (컨텍스트, 시스템 설정 및 사용자 행동)
- 현재 및 과거 프로젝트의 결함 정보
- 결함 사전의 결함 분류
- 적용되는 표준 (예: DO-178B, 등)
- 품질 리스크 (섹션 3.2.1 참고)

각 반복주기 동안, 개발자는 사용자 스토리에서 설명하는 특정한 품질 특성과 관련된 기능/특징의 구현을 위해 코드를 생성하고, 이 코드는 확인 및 인수 테스트를 통해 검증된다. 테스트가 가능하기 위해 인수 기준은 다음과 같은 주제를 반드시 포함해야 한다 [Wiegers13]:

- 기능 동작: 특정 설정에서 입력된 사용자의 행위에 대해 외부에서 관찰 가능한 동작 결과
- 품질 특성: 시스템이 지정된 동작을 수행하는 방법. 특성은 품질 속성 또는 비기능 요구사항을 포함할 수 있다. 일반적인 품질 특성으로는 성능, 안정성, 사용성 등이 있다
- 시나리오 (유스 케이스): 특정 목표 또는 비즈니스 작업을 수행하기 위한 외부 액터 (주로 사용자)와 시스템 사이의 작업 순서
- 비즈니스 규칙: 외부 절차나 제약사항에 의해 정의된 특정 조건 하에서 시스템에서 수행할 수 있는 활동 (예: 보험 청구를 처리하도록 보험 회사에서 사용하는 절차 등)
- 외부 인터페이스: 개발된 시스템과 외부 세계의 연결에 대한 설명. 외부 인터페이스는 서로 다른 유형으로 구분할 수 있다 (사용자 인터페이스, 다른 시스템 인터페이스 등)
- 제약 사항: 개발자의 옵션을 제한하는 모든 설계 및 구현의 제약 사항. 임베디드 소프트웨어를 탑재한 장비들은 일반적으로 크기, 무게, 인터페이스 연결과 같은 물리적 제약을 준수해야 한다
- 데이터 정의: 고객이 복잡한 구조의 비즈니스 데이터에 대해서 각 항목의 포맷, 데이터 종류, 허용되는 값 및 디폴트 값을 기술할 수 있다 (예: US 우편 주소의 ZIP 코드 등)

사용자 스토리 및 이와 관련된 인수 조건 외에도 테스터는 다음과 같은 부가적인 정보를 고려해야 한다:

- 시스템의 의도된 동작 방식 및 사용 방식
- 시스템을 테스트하는데 사용할 수 있는 시스템 접근/사용 인터페이스
- 현재 사용하는 도구 지원의 충분성 여부
- 테스터가 시험을 수행하는데 필요한 충분한 지식과 기술이 있는지의 여부

반복주기가 진행되는 과정에서 테스터들은 종종 추가적인 정보(예: 코드 커버리지 등)의 필요성을 발견하게 되므로 다른 애자일 팀 멤버들과의 협업을 통해 이러한 정보들을 수집해야 한다. 관련 정보는

특정 작업을 완료로 간주할 수 있는지의 여부를 결정하는 역할을 한다. 애자일 프로젝트에서 완료 정의는 매우 중요하며, 이어지는 절에서 논의할 바와 같이 다양한 방식으로 적용된다.

테스트 레벨

각 테스트 레벨은 레벨 고유의 종료 조건을 가지고 있으며 다음 리스트의 예를 종료 조건으로 적용할 수 있다:

- **단위테스팅**
  - 불가능한 경로에 대한 강도 높은 리뷰 및 결정 커버리지 100% 만족
  - 모든 코드에 대한 정적 분석 수행
  - 해결되지 않는 메이저 결함 없음(우선순위 및 심각도를 고려하여 분류)
  - 설계와 구현 시 수용 불가능한 기술적 부채 없음 [존스 11]
  - 모든 코드, 단위 테스트, 단위 테스트 결과 리뷰
  - 모든 유닛 테스트 자동화
  - 중요한 특성에 대한 제약사항에 동의 (예: 성능)

- **통합 테스팅**
  - 크기/복잡도 리스크에 기반한 다수의 긍정/부정 테스트를 포함한 모든 기능 요구사항 테스트
  - 단위 간 모든 인터페이스 테스트
  - 합의한 테스트 강도에 따라 모든 품질 리스크 대응
  - 해결되지 않은 메이저 결함 없음(리스크와 중요도에 따라)
  - 발견된 모든 결함 보고 완료
  - 가능한 모든 회귀 테스트 자동화 및 자동화된 테스트의 공용 저장소 저장

- **시스템 테스팅**
  - 사용자 스토리 및 기능의 엔드-투-엔드 테스트
  - 모든 사용자 페르소나 고려
  - 시스템의 가장 중요한 품질 특성(예: 성능, 안정성, 신뢰성) 커버 완료
  - 가능한 지원되는 환경 구성에 대한 모든 하드웨어 및 소프트웨어를 포함하여 실제 사용 환경에서 테스트 완료
  - 테스트에서 커버하기로 한 모든 품질 리스크 커버 완료
  - 가능한 모든 회귀 테스트 자동화 및 자동화된 테스트의 공용 저장소 저장
  - 발견된 모든 결함 보고 및 가능한 수정 완료
  - 해결되지 않은 주요 결함 없음(리스크과 중요도에 따라 우선 순위화)

유저 스토리 User Story

유저 스토리 완료 조건은 다음과 같다:

- 반복주기에서 선택된 사용자 스토리가 팀에 의해 완료되고, 테스트 가능한 상세한 인수 기준을 가짐
- 인수 테스트를 포함한 사용자 스토리의 모든 요소가 구체화되어 리뷰되고 구현이 완료됨
- 선택한 사용자 스토리의 테스트에 필요한 작업을 식별하고 팀에 의해 추정 완료

기능 Feature

여러 유저 스토리와 에픽을 포함하는 기능에서 완료의 정의는 다음을 포함할 수 있다:

- 인수 기준이 있는 모든 사용자 스토리가 정의되고 고객에 의해 승인 완료
- 알려진 기술 부채 없이 설계 완료
- 알려진 기술 부채 또는 미완성 리팩토링 없이 코딩 완료
- 단위 테스트가 수행되고, 정의된 커버리지 수준 달성
- 기능의 통합 테스트 및 시스템 테스트가 정의된 커버리지 기준에 따라 수행 완료
- 주요 결함 수정 완료
- 릴리즈 노트, 사용자 설명서 및 온라인 도움말 기능을 포함한 기능 설명서 완성

반복주기 Iteration

반복주기에서 완료의 정의는 다음을 포함할 수 있다:

- 반복주기에 대한 모든 기능은 기능 수준의 기준에 따라 준비되고 개별적으로 테스트 완료
- 반복주기의 제약 내에서 해결할 수 없는 중요하지 않은 결함을 제품 백로그에 추가하고 우선순위에 따라 분류
- 반복주기에 대한 모든 기능의 통합을 완료하고 테스트 완료
- 문서의 작성, 검토, 승인 완료

반복주기가 완료되었기 때문에 이 시점에서 소프트웨어는 잠재적으로 출시가 가능하지만, 출시의 모든 반복주기가 완료된 것은 아닐 수 있다.

출시(릴리즈)

여러 반복주기를 포함하는 출시에서 완료의 정의는 다음을 포함할 수 있다:

- 커버리지: 출시의 모든 내용에 대한 모든 관련 테스트 기준이 테스트에 포함되어 있다. 커버리지의 적정성은 복잡성과 크기, 그리고 실패와 관련된 리스크 및 변경 사항에 의해 결정된다

- 품질: 결함 강도(예: 하루 또는 트랜잭션 당 발견되는 결함의 수), 결함 밀도 (예: 사용자 스토리, 노력, 또는 품질 특성의 수에 비례해 발견된 결함의 수), 결함 허용 범위 내에 있는 잠재 결함의 수, 해결되지 않거나 남아있는 결함(예: 심각도 및 우선 순위), 식별한 각각의 품질 리스크와 관련된 리스크의 잔류 수준이 이해하고 허용할 만한 수준인가
- 시간: 미리 정해진 납기에 도달한 경우, 연관된 비즈니스 사항을 고려하여 출시를 결정한다
- 비용: 예상된 수명 주기 비용은 출시 시스템에 대한 투자 대비 수익률을 계산하는데 사용된다 (즉, 계산된 개발 및 유지 보수 비용이 제품의 예상 총 매출보다 상당히 낮아야 한다). 제품 생산 이후에 발생한 결함으로 인해 수명주기 비용의 주요 부분은 종종 유지보수 단계에서 발생하기도 한다

3.3.2 인수 테스트 주도 개발 적용하기

인수 테스트 주도 개발은 테스트 우선 접근법이다. 테스트 케이스들은 해당 사용자 스토리를 구현하기 전에 작성된다. 테스트 케이스는 개발자, 테스터, 업무 대표자[Adzic09]를 포함한 애자일 팀에 의해 생성되고, 수동 또는 자동으로 수행된다. 첫번째 단계로 사용자 스토리를 분석/논의하고, 개발자, 테스터 및 업무 대표자가 요구사항을 작성하는 워크숍을 진행한다. 사용자 스토리의 모든 불완전성, 모호성, 오류가 이 과정에서 보완된다.

다음 단계는 테스트 케이스를 생성하는 것이다. 테스트 케이스 작성은 모든 사람이 참여하거나 테스트 팀에 의해 개별적으로 진행될 수 있다. 테스트 케이스의 작성 방식과 관계없이, 비즈니스 대표자와 같은 제 3 자가 테스트 케이스를 검증한다. 테스트 케이스는 사용자 스토리의 특정한 특성을 기술하고 있는 예제들이며, 애자일 팀이 해당 사용자 스토리를 정확하게 구현하도록 도움을 준다. 예제와 테스트는 동일한 대상을 지칭하며, 이 용어들은 대부분은 동일한 의미로 사용된다. 테스트 케이스 생성 작업은 기본적인 예제와 개방형 질문에서 시작한다.

통상적으로, 첫번째 테스트는 예상했던 대로 실행되면 동작 단계에서 예외 또는 오류 없이 올바르게 동작함을 확인하는 긍정 테스트이다. 긍정적인 경로에 대한 테스트를 완료한 후, 팀은 부정적인 경로에 대한 테스트뿐만 아니라 비기능적 특성도 추가해야 한다(예: 성능, 가용성). 테스트 케이스는 모든 이해관계자들이 이해할 수 있도록 기술되어야 하며, 필요한 사전조건, 예를 들면 입력 및 해당 입력에 대한 출력 값 등을 자연어 문장으로 포함할 수도 있다.

예제는 사용자 스토리의 모든 특성을 포함해야 하며 임의의 내용을 덧붙여서는 안된다. 즉, 예시는 스토리 자체에서 설명하지 않은 측면을 다루지 말아야 한다는 것을 의미한다. 또한, 두 개의 예시가 사용자 스토리의 동일한 특성을 설명해서도 안 된다.

3.3.3 기능 및 비기능 블랙박스 테스트 설계

애자일 테스팅에서 수행되는 많은 테스트는 테스터에 의해 생성되며, 이는 개발자가 프로그래밍 활동을 하는 시기에 동시에 이루어진다. 개발자가 사용자 스토리 및 완료 기준에 따라 프로그래밍을 하는 것처럼, 테스터는 사용자 스토리와 완료 기준에 따라 테스트를 설계한다(섹션 3.3.4 에 설명한대로 탐색적 테스트 및 다른 경험 기반 테스트와 같은 일부 테스트는 테스트 실행 중에 테스트를 설계한다). 테스터는 동등 분할, 경계값 분석 등 기존의 블랙 박스 테스트 설계 기법을 활용할 수 있다. 결정 테이블 테스트 및 상태 전이 테스트를 통해서도 이러한 테스트를 설계할 수 있다. 예를 들어, 경계값 분석은 고객이 구입할 수 있는 항목의 수가 한정되어 있는 경우, 테스트 데이터를 선택하는데 사용될 수 있다.

많은 경우, 비기능적 요구사항은 사용자 스토리와 같이 문서화 된다. 경계값 분석과 같은 블랙 박스 테스트 설계 기법도 비기능적 품질 특성에 대한 테스트를 작성하기 위해 사용될 수 있다. 사용자 스토리에서 성능이나 신뢰성 요구 사항을 포함 할 수도 있다. 예를 들어, 주어진 실행 제한 시간을 초과할 수 없다거나 특정 작업의 실패는 특정 횟수보다 적어야 한다고 명시하는 것이다.

블랙 박스 테스트 설계 기법의 사용에 대한 자세한 내용은 Foundation level 실라버스[ISTQB_FL_SYL] 및 Advanced level_Test Analysis 실라버스[ISTQB_ALTA_SYL]를 참고하자.

3.3.4 탐색적 테스팅과 애자일 테스팅

탐색적 테스팅은 애자일 프로젝트에서 매우 중요한데, 이는 테스트 분석에 사용 가능한 시간과 사용자 스토리의 세부 사항들이 제한되어 있기 때문이다. 최상의 결과를 얻기 위해서는 탐색적 테스팅을 다른 경험 기반 테스팅 기법과 조합함은 물론, 리스크 분석 기반 테스팅, 요구사항 분석 기반 테스팅, 모델 기반 테스팅 및 리그레션 회피 테스팅과 같은 다양한 테스팅 전략을 적절히 혼합해서 활용해야 한다. 테스트 전략 및 이 전략들의 조합과 관련된 자세한 내용은 Foundation level 실라버스를 참조한다.

탐색적 테스팅에서 테스트 설계와 테스트 실행은 미리 준비한 테스트 차터에 기반해 동시에 실행된다. 테스트 차터는 주어진 테스팅 세션 시간 동안 커버해야 할 테스트 조건들을 제공하며, 탐색적 테스트를 진행하는 동안 가장 최근의 테스트 결과 즉, 이전 세션의 테스트 결과에 따라 다음 테스트의 방향을 조정한다. 미리 설계된 테스팅을 수행할 때 사용하는 화이트 박스, 블랙 박스 기법을 테스트 케이스를 설계하는 경우에도 동일하게 사용할 수 있다.

테스트 차터는 다음 정보들을 포함할 수 있다:

- 액터: 시스템을 사용할 것으로 예상되는 사용자
- 목적: 액터가 달성하고자 하는 특정 목적 즉, 테스트 조건을 포함한 차터의 전체 목적

- 준비: 테스트 실행의 시작을 위해 필요한 준비 사항
- 우선 순위: 해당 테스트 차터의 상대적인 중요성으로, 관련된 사용자 스토리 혹은 리스크 수준에 따라 결정함
- 참조: 요구사항(예: 사용자 스토리), 리스크, 그외 다른 정보들
- 데이터: 테스트에 필요한 데이터
- 활동: 액터가 시스템에서 수행할 수 있는 것(예: "슈퍼 사용자로 시스템에 로그온"), 흥미로운 테스트 항목(긍정적 테스트와 부정적 테스트 모두 포함)
- 오라클: 결함인지 여부를 판단할 수 있는 제품의 올바른 동작 결과(예: 화면에 일어나는 캡처와 사용 설명서에 기록된 화면을 비교)
- 변화: 대안 활동 및 활동을 보완할 수 있는 아이디어

탐색적 테스팅을 관리하기 위해 세션 기반 테스트 관리 기법이 사용될 수 있다.
테스팅 세션은 60~120 분 사이의 시간으로 정의되며, 이 시간 동안에는 외부로부터의 어떠한 방해도 받지 않아야 한다. 테스트 세션은 다음 사항을 포함한다:

- 조사 세션(작동 방법에 대한 세부 내용 학습)
- 분석 세션(기능이나 특징 평가)
- 커버리지 확장(예외 또는 경계 케이스, 시나리오, 상호 작용)

테스트의 품질은 대상 제품에 대해 질문할 수 있는 테스터의 능력에 따라 달라진다. 다음과 같은 질문이 포함될 수 있다:

- 이 시스템에서 가장 중요한 것은 무엇인가?
- 어떤 방식으로 시스템이 실패 할 수 있는가?
- 만약 ~ 하면 어떤 일이 생길까?
- …일 때 어떤 일이 발생해야 하는가?
- 고객의 니즈, 요구 사항, 기대가 모두 충족되었는가?
- 시스템 설치(혹은 제거) 시 모든 업그레이드 경로에서 정상적으로 설치(및 삭제)가 수행되는가?

테스트를 실행하는 동안, 테스터는 창의성, 직관, 지식과 모든 역량을 동원해 제품에서 발생 가능한 문제점들을 찾아낸다. 테스터는 또한 테스트 대상, 비즈니스 영역, 소프트웨어 사용 방법, 시스템의 실패 여부 결정 방법에 대한 충분한 지식과 이해를 가지고 있어야 한다.

테스트 중, 다음의 휴리스틱 항목을 적용할 수 있다. 휴리스틱은 경험 기반 테스트를 수행하는 방법 및 결과를 평가하는 방법을 안내해줄 수 있다 [헨드릭슨]. 휴리스틱의 예는 다음과 같다:

- 경계값
- CRUD (쓰기, 읽기, 갱신, 삭제)
- 설정 변경
- 의도치 않은 장애 상황(예: 로그 오프, 종료, 혹은 시스템 재부팅)

테스터는 테스팅 과정을 최대한 문서화해야 한다. 그렇지 않으면 시스템에서 문제가 발견된 경로를 재현하기 어렵다. 다음은 문서에 포함할 유용한 정보의 예시이다:

- 테스트 커버리지: 어떤 입력 데이터가 사용되었고, 어느 만큼의 영역을 커버했는지, 앞으로 테스트 해야 할 부분이 얼마나 남았는지에 대한 정보
- 평가 노트: 테스트하는 동안 관찰한 내용. 시스템과 기능은 안정적인지, 현재 관찰에 따라 다음에 어떤 부분을 테스트하면 좋을 지에 대한 아이디어 등
- 리스크 및 전략: 어떤 리스크가 해소되었는지, 가장 중요한 리스크 중 테스트 안된 부분은 어디인지, 초기 테스트 전략이 지켜지고 있는지, 어떤 변화가 필요한지에 대한 정보
- 문제, 질문, 이상한 점: 예상치 못한 결과, 접근 방식의 효율성에 관한 질문, 아이디어 및 테스트 시도에 대한 우려사항, 테스트 환경, 테스트 데이터, 기능, 테스트 스크립트 또는 테스트중인 시스템에 대한 오해
- 실제 동작: 시스템의 실제 동작에 대한 기록 (예: 비디오, 화면 캡처, 출력 데이터 파일 등)

기록된 정보는 저장되고 특정한 상태 관리 도구(예: 테스트 관리 도구, 태스크 관리 도구, 태스크 보드 등)에 적합한 형태로 요약되어 이해 관계자가 수행된 모든 테스팅의 현재 상태를 쉽게 이해할 수 있도록 해야 한다.

3.4 애자일 프로젝트를 위한 도구들

ISTQB Foundation level 실라버스에 소개된 도구들은 애자일 팀의 테스트와 무관하지 않으며, 실제로 프로젝트에서 테스터들이 사용한다. 모든 도구가 동일한 방식으로 사용되지는 않으며, 몇몇 도구들은 특히 애자일 프로젝트와 더욱 밀접하게 관련되어 있다. 예를 들어, 테스트 관리 도구, 요구 사항 관리 도구, 결함 관리 도구(결함 추적 도구)는 애자일 팀에서 사용될 수 있지만, 일부 애자일 팀은 작업 보드, 번 다운 차트, 사용자 스토리와 같은 애자일 개발과 관련된 기능을 포함하는 도구를 선택하기도 한다 (예: 애플리케이션 라이프사이클 관리 또는 작업 관리 도구). 모든 수준에서 자동화된 테스트 및 관련된 테스트 아티팩트를 관리할 필요가 있으므로 애자일 팀에서 형상 관리 도구는 중요하다.

ISTQB Foundation level 실라버스에 설명된 도구 이외에도 애자일 프로젝트에서 테스터는 이번 장에서 설명하는 도구를 활용할 수 있다. 이 도구들은 애자일 실천방법의 핵심인 팀 협업 및 정보 공유를 보장하기 위해 사용된다.

3.4.1 작업 관리 및 추적 도구

일부 애자일 팀은 각 스프린트가 진행되는 동안 물리적인 스토리/태스크 보드(예: 화이트 보드, 코르크 보드 등)를 사용해 사용자 스토리, 테스트 및 다른 태스크들을 관리하고 추적한다. 다른 팀들은 애플리케이션 수명 주기 관리 및 태스크 관리 소프트웨어를 사용하기도 하며, 여기에는 소프트웨어 태스크 보드가 포함된다.

- 스토리 및 스토리와 관련된 개발 및 테스트 작업을 기록하여 스프린트가 진행되는 동안 태스크가 누락되지 않도록 한다
- 각 태스크에 팀 구성원의 공수 추정치를 기재하고 사용자 스토리를 구현하기 위해 필요한 공수를 자동으로 계산함으로써, 효율적인 반복주기 계획을 수립하도록 한다
- 동일한 사용자 스토리와 연관된 개발/테스트 작업을 묶음으로써 해당 스토리를 구현하는데 필요한 팀의 전체적인 공수를 파악하도록 한다
- 개발자 및 테스터의 작업 완료 상태를 태스크에 반영함으로써, 각각의 스토리, 반복주기 및 전체적인 릴리즈 진행 상태의 스냅샷을 제공한다
- 지리적으로 분산된 팀원을 포함한 모든 이해 관계자들이 각 사용자 스토리, 반복주기, 릴리즈의 현재 상태를 통계, 차트 및 대시 보드를 통해 신속하게 파악할 수 있도록 돕는 시각적 자료를 제공한다
- 태스크와 형상 관리 도구를 연동한다. 이들 형상 관리 도구는 코드 체크인 및 해당 태스크과 관련된 빌드 내역을 자동으로 기록하며, 일부 도구들은 태스크의 업데이트 상태도 기록한다

커뮤니케이션 및 정보 공유 도구

이메일, 문서, 구두 커뮤니케이션 외에도 애자일 팀은 종종 커뮤니케이션과 정보 공유를 강화하는 세 가지 추가적인 도구(위키, 메신저, 데스크탑 공유)를 사용한다.

위키를 사용하면 팀은 프로젝트의 다양한 관점에 대한 지식을 온라인상에서 생성하고 공유할 수 있으며, 다음 정보들을 포함할 수 있다:

- 제품 기능 다이어그램, 기능 관련 토론, 프로토타입 다이어그램, 화이트 보드 상에 기록된 토의 내용의 사진 및 기타 정보
- 팀의 다른 구성원에게 유용할 수 있는 개발 및 테스트를 위한 도구 또는 기술

- 제품 상태와 관련된 지표, 차트 및 대시 보드. 이러한 요소들은 위키가 제품 상태를 자동으로 업데이트 해주는 빌드 서버 및 작업 관리 시스템 등과 연동된 경우 특히 유용하다.
- 인스턴트 메신저 및 이메일 등의 방식으로 팀의 다른 사람들과 공유된 팀 구성원 간의 대화

인스턴트 메시징, 컨퍼런스 콜, 화상 채팅 도구는 다음과 같은 이점을 제공한다:

- 팀 구성원, 특히 지리적으로 분산된 팀원 사이의 실시간 대면 커뮤니케이션이 가능
- 스탠드업 미팅에 지리적으로 분산된 팀 참여 가능
- VoIP 기술을 활용해 통신비를 줄임으로써, 분산된 팀 구성원의 커뮤니케이션을 물리적으로 줄어들게 하는 비용 문제를 줄임

데스크탑 공유 및 캡처 도구는 다음과 같은 이점을 제공한다:

- 분산된 팀에서 제품 시연, 코드 리뷰 및 페어 작업 가능
- 각 반복주기의 끝에서 제품 시연을 캡처하여 팀의 위키에 게시 가능

이러한 도구들은 대면 의사 소통을 대처하는 수단이 아니라 보완하고 확장하는 수단으로만 사용해야 한다.

3.4.3 소프트웨어 빌드 및 배포 도구

본 실라버스의 앞에서 이미 설명한 바와 같이, 소프트웨어를 매일 빌드하고 배포하는 작업이 애자일팀에게는 매우 중요하다. 이 작업을 수행하기 위해서는 지속적인 통합 및 빌드 배포 도구가 필요하며, 이러한 도구들의 장점과 리스크는 1.2.4 절에서 설명되었다. 이는 분산 빌드 도구 및 지속적인 통합 도구의 사용을 필요로 한다. 이러한 도구의 사용으로 인한 장점 및 리스크은 1.2.4 절 앞부분에서 설명되어 있다.

3.4.4. 형상 관리 도구

애자일 팀에서 형상 관리 도구는 소스 코드 저장 및 자동 테스트에 사용될 뿐만 아니라 수동 테스트 및 기타 테스트 산출물 또한 종종 제품 소스 코드와 같은 저장소에 저장된다. 형상관리 도구는 소프트웨어 버전과 테스트 산출물 버전 사이의 추적성을 제공하고, 기록된 정보의 손실을 없애 빠르게 변화에 대응할 수 있게 한다. 버전 관리 시스템의 주요유형은 중앙 소스 제어 시스템 및 분산 버전 제어 시스템을 포함한다. 팀의 크기, 구조, 위치, 타른 도구와의 연동 요구 사항을 바탕으로 특정 애자일 프로젝트에 적합한 형상 관리 도구를 결정한다.

3.4.5 테스트 설계, 구현, 실행 도구

몇몇 도구들은 소프트웨어 테스팅 프로세스의 특정 지점에서 애자일 테스터에 유용하다 대부분의 도구들은 애자일에 특화되어있거나 애자일을 위해서만 만들어진 것은 아니지만, 애자일 프로젝트의 빠른 변화에 대응할 수 있는 중요한 기능들을 제공한다.

- 테스트 설계 도구: 마인드 맵과 같은 도구를 사용해 새로운 기능과 관련된 테스트를 빠르게 설계하고 정의하는 것이 보편화되었다
- 테스트 케이스 관리 도구: 애자일에서 사용하는 테스트 케이스 관리 도구는 팀 전체의 개발 수명 주기 관리 또는 작업 관리 도구의 일부가 될 수 있다
- 테스트 데이터의 준비 및 생성 도구: 데이터 및 데이터의 조합이 많은 애플리케이션을 테스트하는 경우, 애플리케이션의 테스트 데이터를 생성해주는 도구는 매우 유용하다. 이러한 도구는 또한 애자일 프로젝트 도중에 변경이 발생했을 때, 데이터 베이스 구조를 재정의하기 위한 데이터를 생성하는 스크립트의 리팩토링에도 도움이 된다. 이를 통해 변경 사항이 발생한 경우 테스트 데이터를 신속하게 업데이트할 수 있다. 원본 데이터를 초기 입력자료로 사용하는 일부 테스트 데이터 준비 도구는 민감한 데이터를 제거하거나 익명화하기 위한 스크립트를 지원하기도 한다. 또한, 대규모 데이터 입력 또는 출력의 유효성 검증도 가능하다.
- 테스트 데이터 입력 도구: 테스트 데이터를 생성한 후에는 해당 데이터를 어플리케이션에 입력해야 한다. 수동 데이터 입력에는 보통 많은 시간이 소요되고 많은 에러가 발생하므로 데이터 입력 도구를 이용하여 이를 안정적이고 효율적으로 만드는 것이 가능하다. 사실, 데이터 생성 도구의 대부분은 데이터 입력 기능을 지원한다. 지원하지 않는 경우, 데이터베이스 관리 시스템을 사용하여 직접 접근하는 방식(벌크 로딩)도 가능하다.
- 자동화 테스트 실행 도구: 애자일 테스팅에 좀 더 최적화된 테스트 실행 도구들이 있다. 이런 도구들은 행위 주도 개발, 테스트 주도 개발 및 인수 테스트 주도 개발과 같은 테스트 우선 접근 방식을 지원하며, 상용 또는 오픈 소스로 사용이 가능하다. 또한 일부 도구는 테스터 및 비즈니스 관련 인력이 테이블이나 자연어 키워드를 사용하여 시스템의 예상 동작을 표현할 수도 있다.
- 탐색적 테스트 도구: 탐색적 테스트 세션 동안 응용 프로그램에서 수행된 내용 및 로그를 기록하여 테스터 및 개발자에게 도움을 주는 도구이다. 오류가 발생하기 전에 취해진 행동을 기록함으로써 개발자가 결함을 재 발생시키고 분석하는데 유용하다. 궁극적으로 자동화된 회귀 테스트 스위트가 구축되어있는 경우, 탐색적 테스트 세션에서 행동을 기록하는 것이 더욱더 도움이 된다는 것은 이미 증명되어 있다.

3.4.6 클라우드 컴퓨팅과 가상화 도구

가상화는 하나의 물리적 자원(서버)을 다수의 개별 자원으로 활용할 수 있게 해주는 기술이다. 가상 머신 또는 클라우드 인스턴스를 사용하면, 팀은 개발 및 테스트를 위한 대규모 서버 자원을 가질 수 있게 된다. 이는 물리적 서버의 활용과 관련된 개발 지연을 방지하는데 도움이 된다. 대부분의 가상화 도구는 스냅샷 기능을 포함하고 있어 새로운 서버를 구축하거나 서버 복원 작업도 손쉽게 수행할 수 있다. 일부 테스트 관리 도구는 가상화 기술을 이용해 결함 발생 시점에 서버의 스냅샷을 저장함으로써, 테스트와 개발자가 문제점을 공유하고 해당 결함을 조사할 수 있도록 지원한다.

4. 참고자료

4.1 표준

- [DO-178B] RTCA/FAA DO-178B, Software Considerations in Airborne Systems and Equipment Certification, 1992.
- [ISO25000] ISO/IEC 25000:2005, Software Engineering - Software Product Quality Requirements and Evaluation (SQuaRE), 2005.

4.2 ISTQB 문서

- [ISTQB_ALTA_SYL] ISTQB Advanced Level Test Analyst Syllabus, Version 2012
- [ISTQB_ALTM_SYL] ISTQB Advanced Level Test Manager Syllabus, Version 2012
- [ISTQB_FA_OVIEW] ISTQB Foundation Level Agile Tester Overview, Version 1.0
- [ISTQB_FL_SYL] ISTQB Foundation Level Syllabus, Version 2011

4.3 관련 서적

[Aalst13] Leo van der Aalst and Cecile Davis, “TMap NEXT® in Scrum,” ICT-Books.com, 2013.

[Adzic09] Gojko Adzic, “Bridging the Communication Gap: Specification by Example and Agile Acceptance Testing,” Neuri Limited, 2009.

[Anderson13] David Anderson, “Kanban: Successful Evolutionary Change for Your Technology Business,” Blue Hole Press, 2010.

[Beck02] Kent Beck, “Test-driven Development: By Example,” Addison-Wesley Professional, 2002.

[Beck04] Kent Beck and Cynthia Andres, “Extreme Programming Explained: Embrace Change, 2e” Addison-Wesley Professional, 2004.

[Black07] Rex Black, “Pragmatic Software Testing,” John Wiley and Sons, 2007.

[Black09] Rex Black, “Managing the Testing Process: Practical Tools and Techniques for Managing Hardware and Software Testing, 3e,” Wiley, 2009.

[Chelimsky10] David Chelimsky et al, “The RSpec Book: Behavior Driven Development with Rspec, Cucumber, and Friends,” Pragmatic Bookshelf, 2010.

[Cohn04] Mike Cohn, “User Stories Applied: For Agile Software Development,” Addison-Wesley Professional, 2004.

[Crispin08] Lisa Crispin and Janet Gregory, “Agile Testing: A Practical Guide for Testers and Agile Teams,” Addison-Wesley Professional, 2008.

[Goucher09] Adam Goucher and Tim Reilly, editors, “Beautiful Testing: Leading Professionals Reveal How They Improve Software,” O'Reilly Media, 2009.

[Jeffries00] Ron Jeffries, Ann Anderson, and Chet Hendrickson, “Extreme Programming Installed,” Addison-Wesley Professional, 2000.
[Jones11] Capers Jones and Olivier Bonsignour, “The Economics of Software Quality,” Addison-Wesley Professional, 2011.
[Linz14] Tilo Linz, “Testing in Scrum: A Guide for Software Quality Assurance in the Agile World,” Rocky Nook, 2014.
[Schwaber01] Ken Schwaber and Mike Beedle, “Agile Software Development with Scrum,” Prentice Hall, 2001.
[vanVeenendaal12] Erik van Veenendaal, “The PRISMA approach”, Uitgeverij Tutein Nolthenius, 2012.
[Wiegers13] Karl Wiegers and Joy Beatty, “Software Requirements, 3e,” Microsoft Press, 2013.

4.4 애자일 용어

ISTQB 의 키워드들은 각 장의 시작 부분에 제시되어 있다. 일반적인 애자일 용어들은 다음의 잘 알려진 출처의 정의를 참고 했다.

http://guide.Agilealliance.org/
http://whatis.techtargetcom/glossary
http://www.scrumalliance.org/

이 문서에서 애자일과 관련된 친숙하지 않은 단어를 확인하고 싶다면 위 사이트의 활용을 추천한다. 위 사이트들은 이 문서가 릴리즈되는 시점에 활성화되어 있음이 확인되었다.

4.5 그외 리소스

다음의 참고목록은 인터넷이나 혹은 그 외 가능한 정보들을 정리한 것이다. 이 참고자료들이 본 문서를 릴리즈하는 시점에 활성화되어 있음이 확인되었지만, 더 이상 활용 가능하지 않을 수도 있으며, ISTQB 는 이에 대해서 아무런 책임도 없음을 명시한다.

- [Agile Alliance Guide] Various contributors, http://guide.Agilealliance.org/.
- [Agilemanifesto] Various contributors, www.agilemanifesto.org.
- [Hendrickson]: Elisabeth Hendrickson, “Acceptance Test-driven Development,” testobsessed.com/2008/12/acceptance-test-driven-development-atdd-an-overview.
- [INVEST] Bill Wake, “INVEST in Good Stories, and SMART Tasks,” xp123.com/articles/invest-in-good-stories-and-smart-tasks.
- [Kubaczkowski] Greg Kubaczkowski and Rex Black, “Mission Made Possible,” www.rbcs-us.com/images/documents/Mission-Made-Possible.pdf.

5. 색인