목 차

[1. 애자일 테스팅 개념 3](#_Toc391989166)

[1.1. 애자일 테스팅이란? 3](#_Toc391989167)

[1.2. 애자일 테스팅의 원칙 3](#_Toc391989168)

[1.3. 애자일 테스트와 전통적 테스트의 차이점 4](#_Toc391989169)

[1.4. 애자일 테스터에게 필요한 기술 5](#_Toc391989170)

[1.5. 애자일 테스터의 역할 6](#_Toc391989171)

[1.6. 애자일 테스트의 장점 6](#_Toc391989172)

[1.7. 애자일 테스팅의 문제점 7](#_Toc391989173)

[1.8. 애자일 테스팅의 성공 요소 8](#_Toc391989174)

[2. 애자일 테스팅 테크닉 8](#_Toc391989175)

[2.1. 테스트 주도개발(TDD) 8](#_Toc391989176)

[2.2. Integration and System Testing 9](#_Toc391989177)

[2.3. 비기능 요구사항 테스트 9](#_Toc391989178)

[3. 애자일 테스팅 프로세스 10](#_Toc391989179)

[3.1. 개요 10](#_Toc391989180)

[3.2. Roles and Responsiblilties 11](#_Toc391989181)

[3.3. 테스트 전략과 계획 12](#_Toc391989182)

[3.4. Testing During Iterations 13](#_Toc391989183)

[3.5. Testing During Release 14](#_Toc391989184)

[3.6. 테스트 환경 및 Infrastructure 14](#_Toc391989185)

[4. Working on Distributed Teams 15](#_Toc391989186)

[4.1. Testing within ADM for Distributed Agile: 15](#_Toc391989187)

[5. 테스트 자동화 전략 16](#_Toc391989188)

[5.1. 애자일 테스팅 사분면 16](#_Toc391989189)

[5.2. 테스트 자동화 피라미드 17](#_Toc391989190)

[5.3. 테스트 자동화 계획 18](#_Toc391989191)

[5.4. 테스트 자동화 프레임워크 18](#_Toc391989192)

[5.5. 테스트 자동화 대상 선정 18](#_Toc391989193)

[5.6. 테스트 자동화 도구 18](#_Toc391989194)

[6. Reference 19](#_Toc391989195)

# 애자일 테스팅 개념

## 애자일 테스팅이란?

애자일 가치와 원칙을 따르는 테스팅 방법으로 “ 개발”을 “테스팅”의 “고객”으로 다룬다. 애자일 테스팅은 엄격하게 정의된 테스트 절차를 강조하지 않으며 새롭게 개발된 코드에 대해 최종 고객의 관점에서 품질이 달성될 때까지 반복적으로 테스트 하는데 중점을 둔다.

## 애자일 테스팅의 원칙

### 테스트는 전체 팀의 책임

* 품질과 테스트에 대한 책임이 테스터 뿐만 아니라 전체 팀의 팀원에게 부여된다. 각 팀의 팀원은 QA안에서 전담 역할을 맡고 있으며 QA를 테스터에게 돌려서는 안된다.

### 반복적이고 점진적인 테스트

* 테스팅은 지속적으로 이루어져야 하며 스프린트 안의 모든 개발 활동에 대한 직접적인 후속조치이다.

### 테스트가 먼저

* 보다 일찍 그리고 완전하게 테스트하는데 중점을 두며, 빠른 피드백을 얻기 위해 테스트 자동화와 지속적인 테스트, 그리고 지속적인 통합(CI)을 통해 실현된다.

### 비즈니스적 가치에 중점

* 테스트 사양에 대한 준수 여부 보다는 최종 사용자의 비즈니스적 가치와 품질에 중점을 둔다.

### 협력과 의사소통

* 엄격한 프로세스와 잘 정의된 테스트 절차 보다는 협력과 의사소통에 중점을 둔다.

### 간소함

* 문서 작성시 간소하고 간결하며 경량적 접근방법을 사용하여 상세하고 서술적인 테스트 문서를 대신한다.

### 자동화

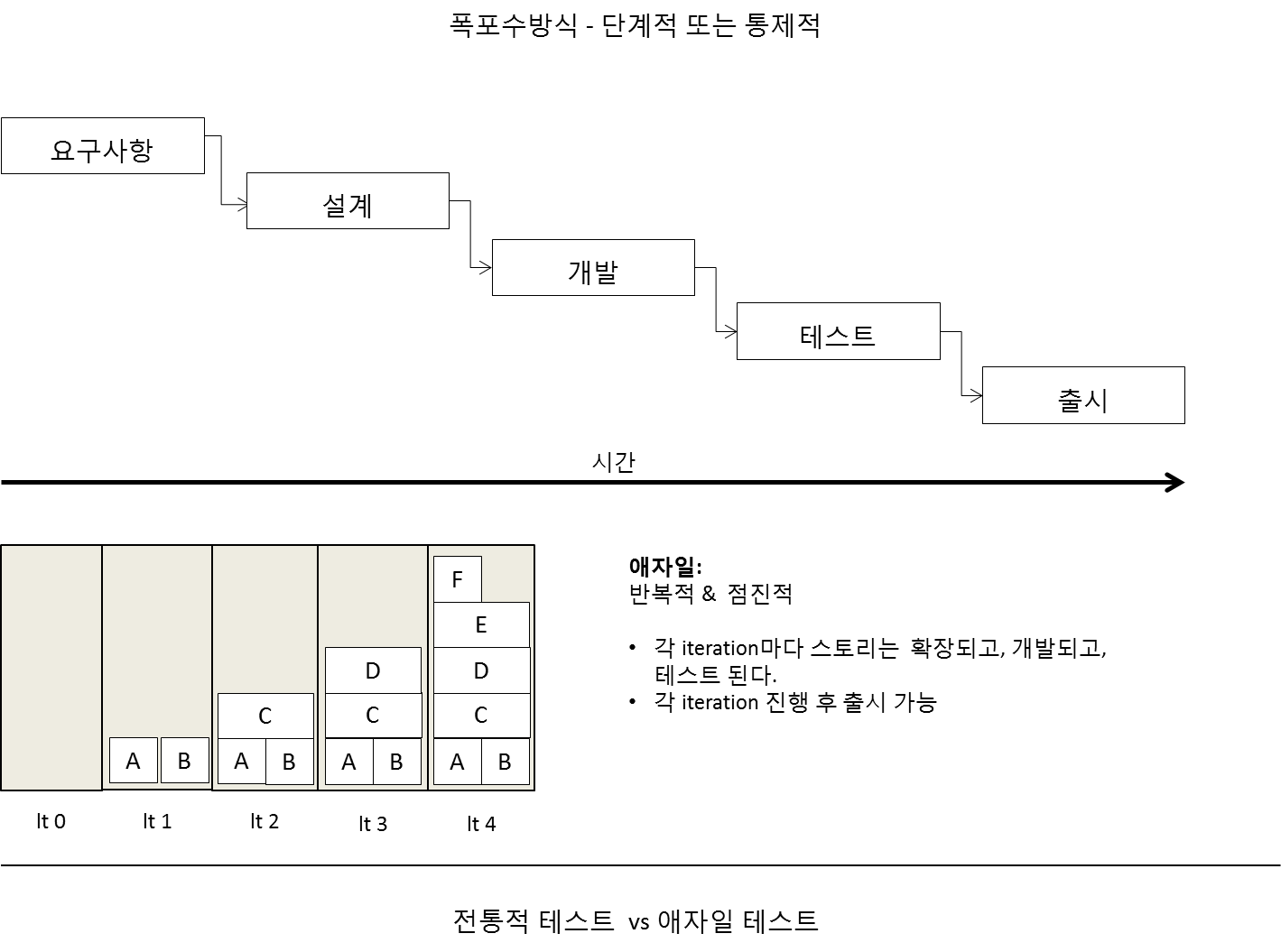
* 반복적이고 점진적인 출시를 위하여 수동 테스트 보다는 자동화 테스트에 중점을 둔다.

### 리스크 기반

* 테스트의 가치와 깊이는 기능 컴포넌트에 대한 비즈니스 가치와 중요도를 기반으로 평가한다.

## 애자일 테스트와 전통적 테스트의 차이점

|  |  |
| --- | --- |
| 전통적 테스트 | 애자일 테스트 |
| 테스트가 주로 끝 부분에서 수행됨 | 테스트가 각 스프린트 안에서 반복적으로 수행됨 |
| 팀간 의사소통, 문서화, 업무인계가 공식적임 | 상호 대화와 의사소통이 대량 발생하므로 항상 공식적이진 않음 |
| 자동화는 선택사항 | 요구되는 자동화 수준이 높다 |
| 테스트는 요구사항 관점에서 수행됨 | 테스트가 고객 관점에서 수행됨 |
| 상세한 명세서와 테스트 계획서가 유지됨 | 유저스토리와 가벼운 테스트 계획으로 낮은 관리 비용 |
| 테스터는 테스트 팀의 일원 | 테스터는 개발/스크럼 팀의 일원 |



## 애자일 테스터에게 필요한 기술

* 팀원과 이해당사자들에게는 긍적적이고 해결지향적인 자세를 갖는다.
* 상품에 대해서는 비판적, 품질지향적, 회의적 사고방식을 견지한다.
* 이해당사자로부터 능동적으로 정보를 수집해야 한다. (전적으로 설계 문서에 의존하지 않는다.)
* 테스트 결과, 진척도 그리고 상품의 품질에 대해 정확하게 평가하고 보고할 수 있어야 한다.
* 테스트 해야하는 유저 스토리, 특히 인수 기준에 대해 효과적으로 정의할 수 있어야 한다.
* 팀 안에서 프로그래머들과 그리고 다른 팀 구성원들과 함께 쌍으로 협력하며 작업할 수 있어야 한다.
* 테스트케이스를 교체, 추가, 개선시킴으로써 변경사항에 빠르게 대응할 수 있어야 한다.
* 스스로 업무를 조직하고 계획할 수 있어야 한다.

## 애자일 테스터의 역할

### 테스터는 스크럼 팀의 일원이다.

* 애자일은 QA를 “전체 팀의 책임”으로 본다. 이상적으로는 테스팅/QA를 스크럼 팀의 각 개인들이 책임지는 것이다. 애자일 팀에서는 테스터와 프로그래머, 그리고 기타 역할의 팀원 사이의 경계가 불분명하다.
* 몇몇 애플리케이션은 스크럼팀의 외부에서 테스트를 수행해야 한다. 예를 들면 보안 테스트나 성능테스트와 같이 특별한 기술이 필요할 경우 외부의 전문가를 스크럼팀에 추가할 수도 있다. 어떤 상황에서는 전문 팀이 꾸려져야 한다. 전문가 또는 전문팀은 스크럼팀과 협업을 하게 된다. 하지만 여전히 “테스트는 맨 나중에” 방식은 허용되지 않는다.

### 품질을 위해 “코치”와 “대사”의 역할을 한다.

* 애자일 테스터는 조기 단계에서 기능 요구사항과 유저스토리에 대한 검증을 수행 한다. (Test-First)
* 테스터는 품질보증을 위한 “대사”역할을 한다.

### 각 스크럼 팀에 분산배치된 테스터들의 지식 교환

* 특정한 테스트 기술과 사상을 수용하는 것은 자동적으로 되는 것이 아니다. 팀원들에게 테스트 방법론 교육을 위한 브라운 백 세션(brown-bag session), 코칭 그리고 CoP(Community of Practices)등이 필요하다.

## 애자일 테스트의 장점

* 프로젝트의 매우 이른 시점에서 치명적 결함이 식별되고 해결된다.
* 반복적 테스트 주기로 인하여 실제 사용가능한 기능들이 신속하게 출시된다.
* 테스트가 완료된 데모 소프트웨어를 통해 고객으로부터 더 자주 피드백을 받을 수 있다.
* 테스터에 의한 광범위한 프로젝트 비전과 통찰력이 전체 품질을 향상시킨다.
* 진척도, 비용 및 품질에 대한 높은 수준의 투명성
* 조기 위험 기반 테스트

## 애자일 테스팅의 문제점

### 통제적 환경에서의 애자일 테스팅:

* 통제적인 환경에서는 높은 수준의 재현 가능한 테스트 커버리지를 제공하기 위해 확실한 계획과 추적성, 테스트 활동에 대한 검증 가능성, 그리고 매우 체계적인 테스트 설계가 요구된다.

### 경량 문서화:

* 스프린트 기간이 짧기 때문에 테스트 팀이 테스트 스크립트를 개발하고 유지보수할 충분한 시간이 없다.
* 상세 테스트 계획 부족 : 요구사항 변경에 따라 테스트 계획도 자주 업데이트 되어야 하므로 단순하게 유지된다.

### 회귀 사이클의 증가

* 회귀테스트의 범위가 각 스프린트가 끝날 때마다 증가함에 따라 회귀테스트 시간도 같이 증가하게 됨

### 잦은 요구사항 변경

* 애자일에서는 요구사항의 변경을 환영하기 때문에 테스터는 요구사항의 변경에 따라 테스트 케이스의 변경이 필수적이다.
* 테스트 데이터의 잦은 변경으로 인한 극단적인 비용 증대를 피하기 위해서 테스트 설계시 특별한 고려가 필요하다.
* 요구사항 변경에 따라 테스트 케이스를 변경하는 것 보다 회귀테스트의 변경에 필요한 공수가 더 많다.

### 전체 시스템에 대한 통찰력 감소

* 테스팅이 스프린트마다 이루어짐으로 인해 시스템 전체를 바라보는 관점을 잃을 수 있다.

### 테스터에게 요구되는 다른기술

* 짧은 스프린트 기간으로 인해 대부분의 테스트는 자동화되어야 한다.
* 테스터에게 자동화 툴에 대한 기술이 요구된다.

## 애자일 테스팅의 성공 요소

### 요구사항 관리

* 요구사항 관리와 테스트 케이스 추적관리를 위한 애자일 도구 필요

### 변경관리

* 변경사항에 대하여 상호 신뢰적이고 신속한 커뮤니케이션
* 구조화 되고, 가독성 높으며, 모듈화된 테스트 스크립트
* 변경 영향도 분석을 위한 도구, 기술, 템플릿

### 지능적인 지속적 통합(CI) 설정

* 코드를 최적화하고 초기에 결함을 발견하고 품질을 보장하는 행위가 포함된다.
* 자동화 기능/UI 테스트를 통합하거나 제외시키는(통상적으로 시간이 오래 걸리므로) 등 다양한 테스트 스위트를 선택적으로 적용하여 최적화되고 빠른 실행이 가능하도록 설정한다.

### 커뮤니케이션 관리

* 애자일 프로젝트에서 커뮤니케이션 계획의 열쇠는 단순성의 원리를 따르는 것이다.
* 문서화를 위한 문서 작성을 하지 않는다.
* 이해당사자가 정말로 무엇이 필요한지를 알아내어 가급적 신속하고 단순하게 제공한다.
* 팀에서 이미 만들고 있는 본래의 정보 소스를 활용한다. (태스크 보드, 번다운 차트, 아키텍처 설명등)
* 의사 결정이 필요한 경우 문서를 작성한다.

# 애자일 테스팅 테크닉

애자일 테스팅 활동은 목적과 가치에 따라 다양한 종류의 테스트로 분할된다. 테스트 타입은 종종 다음의 범주로 세분화된다:  
프로젝트 팀의 개발을 지원하는 테스트, 비즈니스 관점의 품질 검사 테스트, 제품 비평 테스트, 소프트웨어와 배포플랫폼간 관계 검증 테스트

## 테스트 주도개발(TDD)

TDD는 소프트웨어 설계와 구현을 추진시켜주는 애자일 방법론의 중대한 구성요소임. TDD는 기본적으로 설계 행위이지만 코드 품질을 검사하는 자동화된 unit과 component테스트를 제작함으로써 테스팅 목표를 지원한다.

### 단위테스트, 컴포넌트 테스트

* 개발자에 의한 개별 소프트웨어의 유닛과 관련 컴포넌트에 대한 테스트는 소프트웨어의 제작 결함을 발견하는데 있어서 대단히 중요하다.

### 유저스토리 테스트

* 유저스토리 테스트는 애자일프로젝트 안에서 성공적인 소프트웨어의 개발을 위해 매우 중요하다.이 테스트는 종종 위의 테크닉에 의해 수행되지만 필요에 따라 다른 방법으로도 수행될 수 있다.

## 통합 테스트와 시스템 테스트

### 구조적 테스트(Structured Testing)

* 구조적 테스트는 산출물의 구조와 요소에 대한 팀의 지식에 근거하여 스토리 또는 비즈니스 프로세스 상에서 수행될 부가 테스트의 메커니즘을 제공한다.

### 탐색적 테스트(Exploratory Testing)

* 탐색적 테스트는 산출물에 대한 테스터의 직관과 지식에 근거하여 스토리 또는 비즈니스 프로세스 상에서 수행될 부가 테스트의 메커니즘을 제공한다.

## 비기능 요구사항 테스트

### 로드 테스트와 성능 테스트

* 로드와 성능테스트는 소프트웨어가 언제든지 응답시간, 가용성, 계산속도와 같은 요구사항을 반드시 충족할 수 있도록 하여 주는데 있어 중요하다.

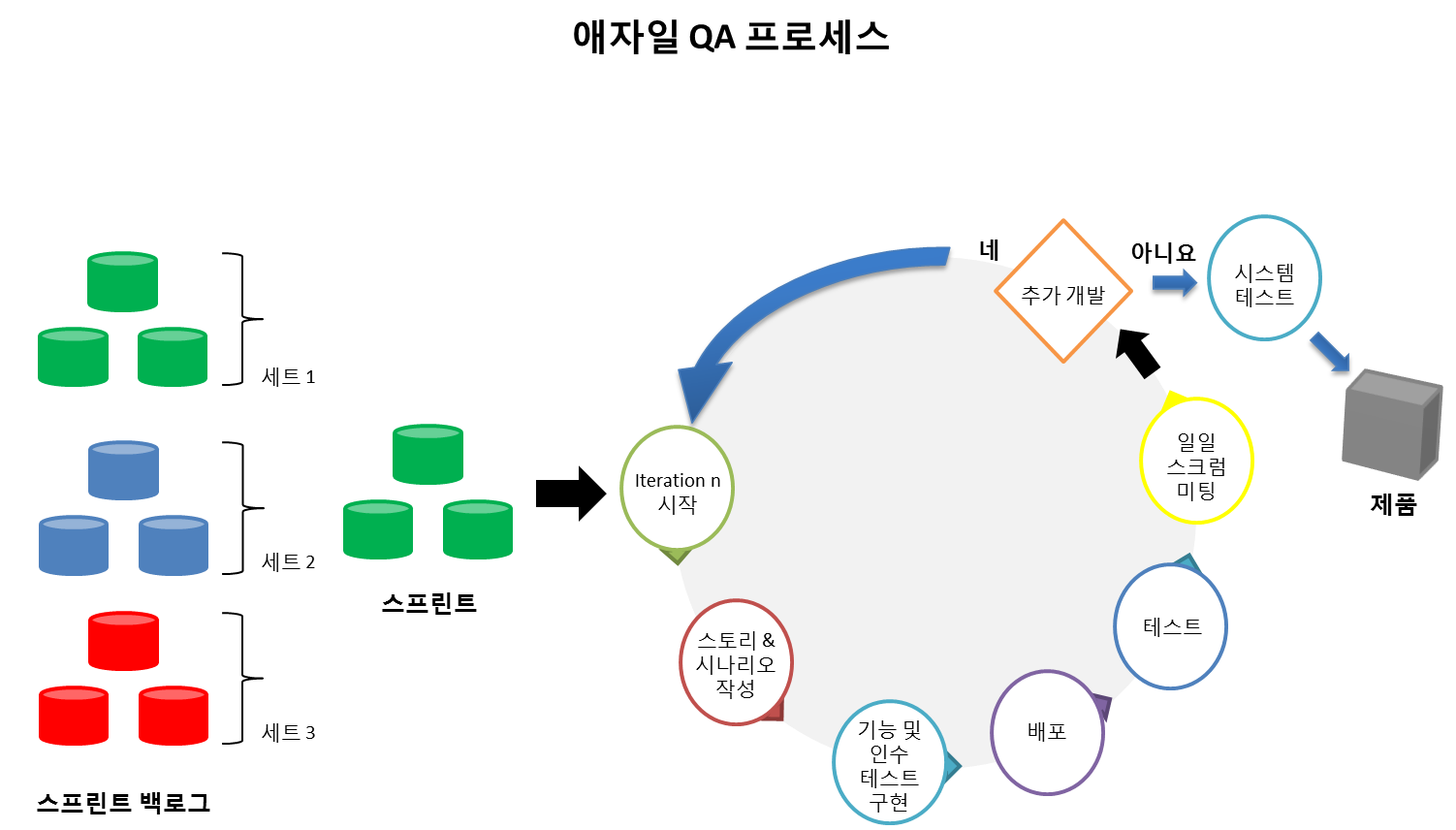
### 보안 테스트

* 보안테스트는 소프트웨어의 취약성으로 인해 중대한 비즈니스 결과를 초래하지 않도록 하는데 있어 중요하다.

# 애자일 테스팅 프로세스

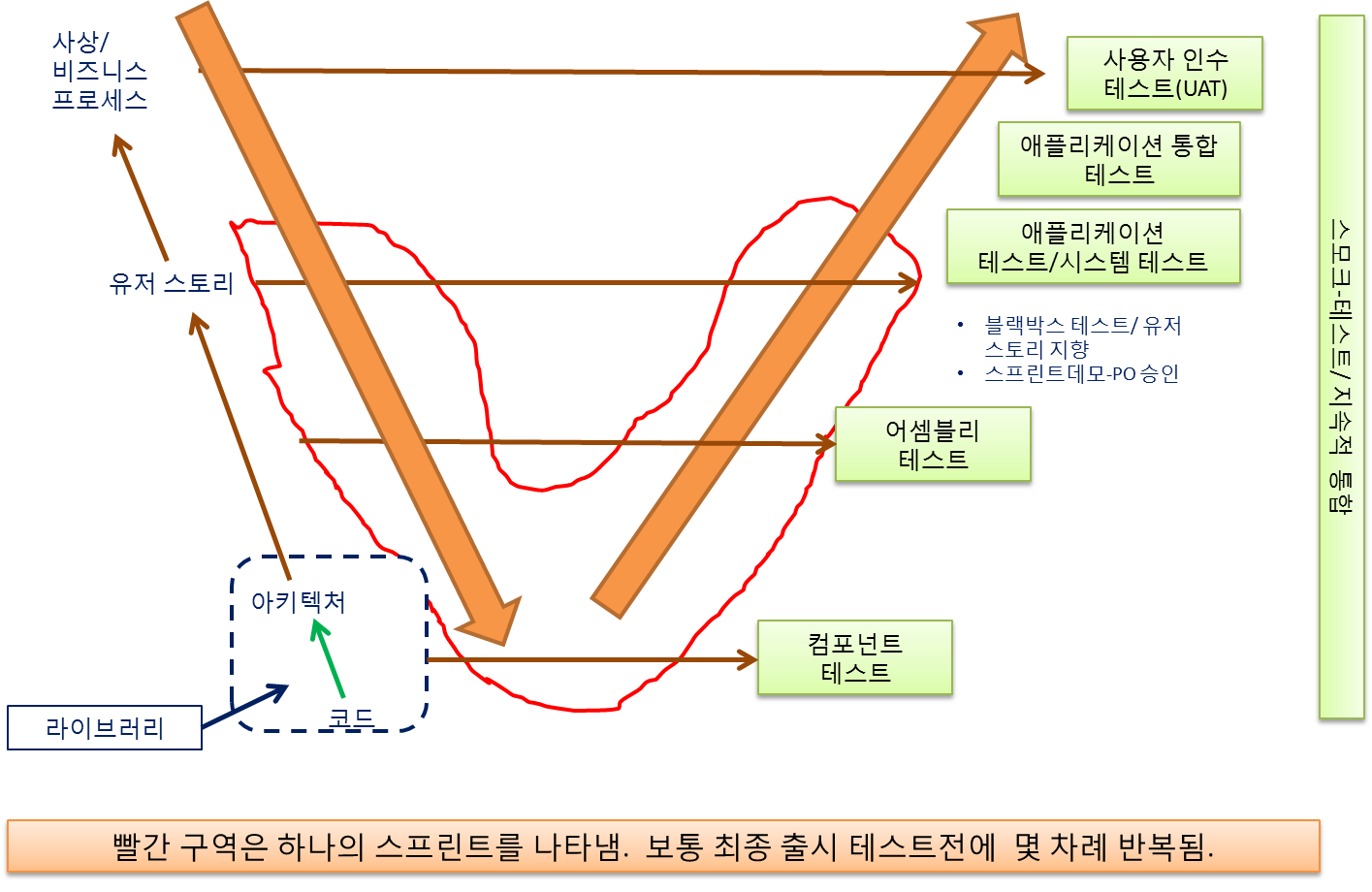
## 개요

### 애자일 QA 프로세스



* 프로세스 스텝들과 테스트 단계는 전통적 테스트 방법론과 유사하다.
* 프로세스 스텝들은 더 짧고, 반복적이며, 개발 사이클 주기와 완전히 통합된다.
* 대형 애자일 프로젝트에서는 릴리즈 테스트 또는 경화 단계로 불리는 특별한 테스트 단계가 도입되었다.
* 설정관리, 문서화 및 추적성 결함 관리,의사소통 오류관리 같은 비용절감을 위한 프로세스들 뿐만 아니라 , 리스크 관리 같은 전통적 프로젝트의 관리 프로세스들도 테스트에 영향을 미친다.

### V-모델



* V-모델의 테스트 레벨과 각 스테이지의 다섯가지 활동들은 애자일 테스트 방법론의 일부이다.
* 테스트 방법론과 적용방안, 측정방법, 기반구조 같은 기초 절차들은 초기의 세부 활동 (계획, 준비, 분석 또는 설계 같은) 안에 포함되어 준비된다.
* 애자일 프로젝트에서는 이와 같은 활동, 특히 테스트 커버리지 나 유저스토리당(스프린트나 릴리즈 단위가 아닌) 허용 오류 수 같은 DoD 기준이 프로젝트의 시작 단계 또는 스프린트 0에서 이루어진다.
* 테스트의 설계와 구현 그리고 평가 활동은 스프린트 안의 컴포넌트 테스트, 컴포넌트 통합 테스트, 그리고 시스템 테스트 수준에서 이루어진다. 비 애자일 프로세스와 대비하여 이러한 테스트들은 애플리케이션 개발과 밀접하게 연결되어 있으며 테스트 자동화에 강력한 초점을 맞추어 짧은 반복 주기로 시행된다.
* 마지막 단계는 각 스프린트의 끝과 릴리즈 전에 일어나는 최종 공식 테스트 완료 단계이다.
* 대형 프로젝트에서는 스프린트 기간 동안 인터페이스들이 비즈니스 프로세스에 따라 지속적으로 구현되지 않았을 가능성이 있기 때문에 릴리즈 전에 시스템 통합 테스트를 위한 추가 시험 기간이 권장된다. 이것은 또한 스프린트 테스트들에 대한 추가적인 안전 그물망 역할을 한다.

## 역할과 책임

### 팀 기반의 테스트 방법론

* 애자일 프로젝트 기간동안 테스트는 팀 기반이며 모든 팀 구성원들이 일정수준 테스트를 지원하는 것이 보통이다.

### 테스트에서 일반적인 PO의 역할

* 일반적으로 PO는 인수 기준에 대한 지침을 제공하고, 스토리가 목표 달성을 위해 의도하는 바를 예를 들어 제공하는 테스트 케이스를 작성하기도 한다.

### 테스트에서 일반적인 개발자의 역할

* 소프트웨어 개발자들은 개발 프로세스 안에서 다양한 수준의 테스트를 구현하고, 자동화하고 수행한다. TDD와 ATDD는 이러한 테스트가 설계와 개발이 개선되도록 한다.

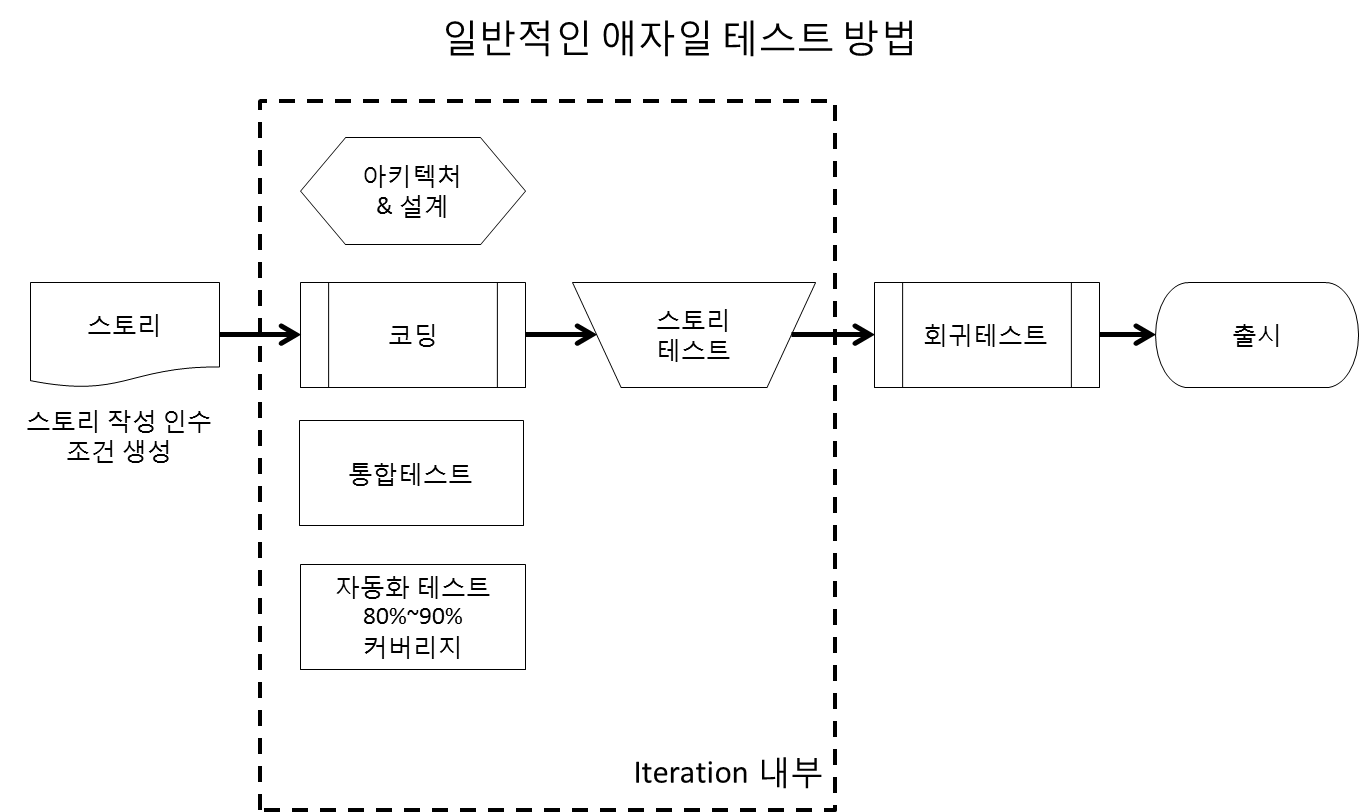
### 테스트에서 일반적인 테스터의 역할

* 소프트웨어 테스터는 일반적으로 PO와 개발자들과 함께 모든 레벨에서 수행되는 테스트를 계획하고, 실행하고 보고한다. 테스터는 종종 유저스토리와 비즈니스 프로세스 테스트 케이스의 작성 및 탐색적 테스트를 수행할 책임이 있다. 테스터는 개발자 또는 테스트 자동화 전담팀과 함께 자동화된 테스트를 개발하는데 참여한다.

### 테스트에서 일반적인 고객의 역할

* 고객은 일반적으로 스토리와 인수 기준을 정의하며 릴리즈를 하기 전 또는 연속적인 iteration후 의미 있는 시점에서 사용자 인수테스트(UAT)를 수행함으로써 테스트를 지원한다.

## 테스트 전략과 계획



### 테스트 계획 문서 경량화

* 애자일에서는 테스트 계획시 작업을 완료하는데 필요한 최소한의 문서만 제공하는 것을 목표로 한다.

### 스토리 추정

* 스토리의 범위(사이즈)는 애자일 팀이 지속 가능한 개발 페이스를 유지하면서 목표 속도가 달성될 수 있도록 추정되어야 한다.

### Consider using Risk-Based testing위험 기반 테스트의 고려

* 테스트 케이스는 요구사항의 우선 순위에 따라 추가, 삭제, 변경될 수 있다. 테스트 케이스는 요구사항의 변경을 충분히 수용할 수 있도록 유연해야 한다.

### Create smart automation test scripts테스트 자동화 스크립트 생성

* Create automation scripts in terms of re-usable rather than one monolithic script. As the automated test script will have to be revisited in each sprint, it is essential to keep the scripts simple
* 재사용이 가능한 자동화 스크립트를 작성한다. 자동화 스크립트는 각 스프린트마다 재사용할 수 있도록 간소하게 유지해야 한다..

### 테스트 작업 계획

* Allocate time for re-factoring test scripts  
  Add testing perspective to the selection of User Stories.
* 테스트 스크립트의 리팩토링 시간을 할당한다.
* 유저스토리 채택 과정에 테스트 관점을 포함시킨다.
* Re-use Unit as well as Functional tests for Regression  
  As far as possible try to reuse the test cases for unit as well as functional testing
* 회귀 테스트시 기능 테스트 뿐만 아니라 단위테스트도 재사용한다.
* Have a separate test phase just before the release  
   It is a best practice to have a test phase just before the release to ensure completeness of testing and optimization of the entire product
* 테스트의 완결성과 제품 전체의 최적화를 보장하기 위하여 출시 전 별도의 테스트 단계를 둔다.

### 인수 기준

* 스토리 인수 기준은 스토리의 “완료”를 정의함에 있어 대단히 중요하며 ATDD의 스토리 테스트시 개발해야 할 테스트 유형에 대한 가이드를 제공해 준다.

## 반복(Iteration)기간 중의 테스트

### Developer/Tester Interactions

* In agile, developers and testers work hand-in-hand to build and test stories.

### Tester/Customer Interactions

* In agile, testers and customers work hand-in-hand to make sure acceptance criteria and acceptance tests are complete and appropriate for the stories that will be implemented and tested during each iteration.  
  Defect tracking and management, the amount of defect tracking that is performed during an agile project depends upon what works best for the team.

### Results reporting

* Test reporting during agile projects depends upon what works best for the team.

### Test metrics

* Metrics collected to support test completeness and release readiness decisions

### Regression tests

* Automated regression tests are essential to reducing the cost of change and providing real-time feedback during the development process.

### Iteration wrap-up

* Wrap-up activities during iteration include a product demo, retrospective, and sometimes a User Acceptance Test.

## Testing During Release

### Definition of a release/end game

* A release process (aka “end game”) is performed whenever a decision has been made to release software to customer(s).

### User Acceptance Testing (UAT)

* User Acceptance Testing is used within agile to gain customer feedback on a working piece of software before its release.

### Staging Environment Testing

* A staging environment is often established to support testing of an application within an environment that closely resembles production.

### Post-release Testing

* Testing after software release typically consists of testing “hot fixes” for critical defects identified in the field and on-going testing of bug fixes not fixed prior to release.

## 테스트 환경 및 Infrastructure

### Typical environments for test

* Multiple environments are often necessary to support testing activities during iterations and the release process

### Virtualization

* Virtualization provides a mechanism (often automated) to support effective test environment setup, test execution, and test environment teardown during a testing process.

### Testing the proper build

* As builds are constantly being generated during an agile process, testing the proper build is critical to an effective testing process

### Test data management

* Effective test data management is essential to all aspects of agile testing as the ability to select appropriate test data, set this data up, perform testing upon it, and reset any resulting changes is critical to an effective testing process.

# Working on Distributed Teams

## Testing within ADM for Distributed Agile:

* Initiation: High-level Test Plan for Integration, Performance and UAT.
* Sprints: Planning and Execution of Unit, Regression and End-to-End product Test.
* Test: Execution of Integration, Performance, End-to-End Test and UAT tests.

### Within Each Sprint

* Component Testing
* Assembly Testing
* Regression Testing
* Application Product Testing

### Outside the Sprints

* Integration Testing
* Performance Testing
* User Acceptance Testing

# 테스트 자동화 전략

## 애자일 테스팅 사분면



### 1사분면

* “White Box” 테스트의 한 종류
* 개발자에 의해 수행됨
* 단위 테스트는 프로그램 디자인 문제에 주목하며, 작은 조각의 코드가 제대로 동작하는가를 검증한다. 컴포넌트 테스트는 시스템 디자인 문제에   
  주목하여 모든 컴포넌트들이 제대로 동작하는지를 검증한다.

### 2 사분면

* “Black Box/Grey Box”테스트의 한 종류
* 기능 요구사항 테스트를 포함함
* 추가적인 영역에서 특수한 전문지식을 가진 테스터에 의해 수행
* 반복적으로 수행
* 회귀 테스트는 자동화 할 것을 권고함

### 3사분면

* “Black box”테스트의 한 종류
* 비즈니스 기능성 테스트를 수반한다.
* 테스터에 의해 수행
* 스프린트 마지막에 고객에게 데모를 수반한다.
* 테스트 주기에 있어 가능한한 빨리 시작해야 한다.

### 4사분면

* “White Box”테스트 후 수행되는 “Black Box”테스트
* 비기능적 요구사항에 대한 테스트를 수반한다.
* 성능과 보안 요구사항 이슈를 다루는 전문 테스터에 의해 수행된다.
* 테스트의 실행은 우선순위와 테스트 환경의 가용성에 달려 있다.

## 테스트 자동화 피라미드

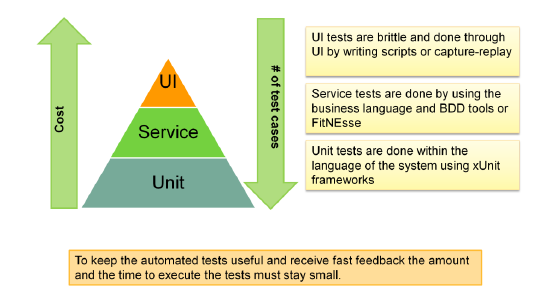


Figure . 마이크 코언(Mike Cohn)의 테스트 자동화 피라미드

* Testing UI is brittle, expensive, and time consuming. Additionally, testing via UI is partially redundant (think about how many times a suite of tests will test the user interface or things which were tested before on lower levels).
* The term ‘Service layer’ is used in a more common manner than within the SOA environment. It is defined as a ‘method’ called by the UI or which directly delivers data to the UI (UI has the function of displaying it smoothly).
* The middle layer of Service testing has been ignored by organizations resulting in incorrect test automation efforts.
* A very broad basis is created by a solid basement of Unit tests.
* Re-use parts of unit as well as functional tests for Regression testing.

## 테스트 자동화 계획

* 테스팅에 전문성을 보유하지 않고서 자동화 테스팅 툴을 활용하는 것은 좋은 방법이 아님
* Mike Kelly에 의하면 목적에 부합하는 테스트 프로젝트를 수행하기 위해서 테스팅 리스크와 범위 등에 대한 테크닉을 익히고 익숙해진 다음 자동화 테스팅으로 전환하는 것이 필요하다고 밝힘
* 접근방법, 도구, 프로젝트 적용시기를 정의한다.

## 테스트 자동화 프레임워크

* 프레임워크는 자동화하는 다양한 종류와 레벨의 테스트에 대한 infrastructure를 제공한다.

## 테스트 자동화 대상 선정

* It is typically infeasible and not cost effective to automate all tests that are created and/or run. Supporting process  
  Test automation is performed at various points during agile project iterations and release cycles. When test automation is performed and for what purpose must be understood.

## 테스트 자동화 도구

* 애자일에서는 자동화에 초점을 맞추기 때문에 도구의 사용은 필수적이다.

|  |  |  |
| --- | --- | --- |
| 목적 | 도구명 | TMON 선정 도구 |
| 테스트 관리 | HP Quality Center, (Mingle?) | JIRA |
| 오류 추적 | JIRA, TFS, Bugzilla | JIRA |
| 소스 코드 관리 | Git | Git |
| 단위테스트 | JUnit | JUnit |
| 지속적통합 | Jenkins | Jenkins |
| 릴리즈 관리 | GO | GO |
| UI 테스트 | Selenium WebDriver, Sikuli, QTP | Selenium WebDriver |
| BDD | Cucumber | - |
| A-TDD | FitNesse | FitNesse |
| 성능 테스트 |  |  |
| 정적 코드 검사 | PMD | PMD |
| 공동작업 | Sharepoint, Confluence | Confluence |

# Reference

* <http://blog.testing-whiz.com/2012_01_01_archive.html>
* Continuous Integration – Martin Fowler  
  <http://martinfowler.com/articles/continuousIntegration.html>
* 테스트 자동화에도 불구하고 프로젝트가 실패하는 4가지 공통요인
* ISTQB:Agile Tester Extension Syllabus  
  <http://www.istqb.org/downloads/syllabi/agile-tester-extension-syllabus.html>