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

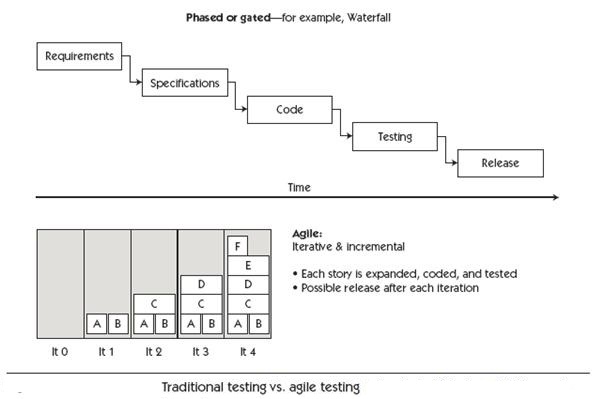
* 반복적이고 점진적인 출시를 위하여 수동 테스트 보다는 자동화 테스트에 중점을 둔다.

### 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

* 팀 멤버들이나 이해당사자들과 긍정적이고 해답 중심적 이어야 한다
* 비판적, 질 중심과 회의적 사고방식을 보여줘야 한다
* 상술 문서에 의존하는 것보단 정보를 이해당사자들에게서 활발하게 얻어야 한다
* 테스트 성적, 테스트 진척과 상품 재질을 정확하게 평가하고 보고해야 한다
* 고객 담당자들 및 이해당사자들과 테스트 가능한 유저 스토리들을 효과적으로 정의해야 한다. 인수 기준을 각별히 신경 써야 한다.
* 팀 내에서, 프로그래머들과 다른 팀 구성원들과 쌍으로 협력하며 작업 해야 한다
* Respond to change quickly, including changing, adding, or improving test cases
* 자신의 작업을 계획 및 구성 해야 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.
* Due to the frequent change in test data, test design needs special considerations to avoid dramatic increase in costs.
* Effort required for regression testing is more as the test cases change with changing requirements.

### 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

* Reliant and fast communication of changes
* Well structured, easy readable, and modularized test scripts
* Tools, techniques, and templates to impact analysis of change

### Smart Continuous Integration Setup:

### Communication Management:

# 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.

### Unit Testing, Component Testing

* 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.

## Integration and System Testing

### 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

### Load and Performance Testing

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

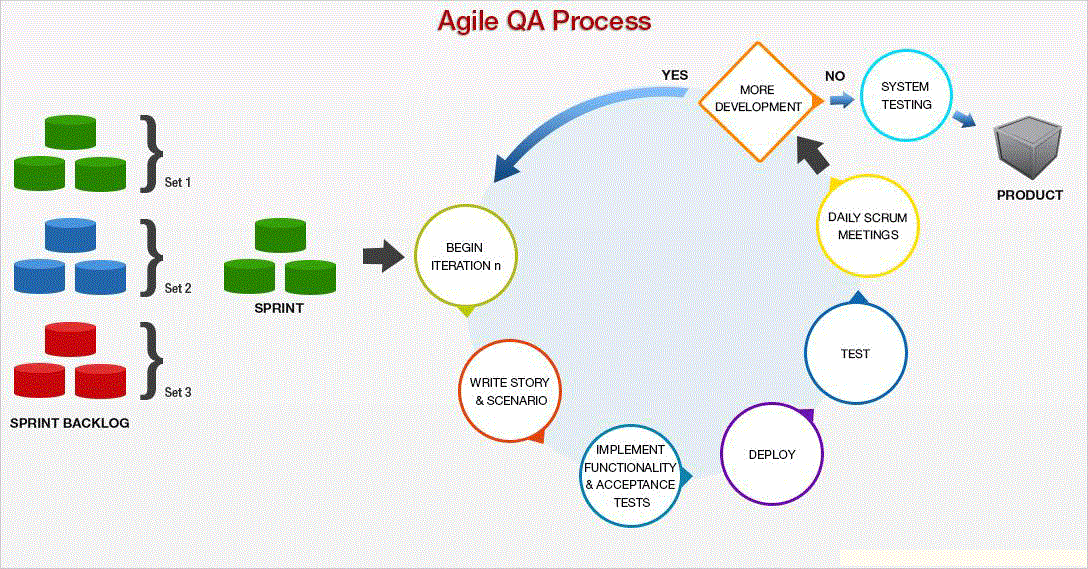
### 보안 테스트

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

# 애자일 테스팅 프로세스

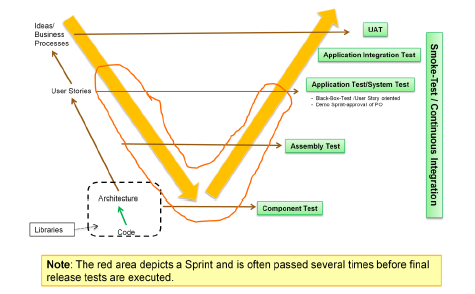
## 개요

### 애자일 QA 프로세스

****

* 프로세스 스텝과 테스트 단계는 전통적 테스팅과 유사하다
* 프로세스 스텝들은 짧고, 반복적이고 개발생명구기와 완전히 통합됐다
* 큰 애자일 프로젝트들을 위해서 출시 테스트 혹은 경화 단계라는 특수 테스트 단계가 설립됐다
* 구성 관리, 문서에 대한 오류 관리, 추적과 오류의 통신 등의 비용 절감 프로세스뿐만 아니라 위험성, 관리 및 보고 등의 전통적 프로젝트 관리 프로세스들은 테스트에 영향을 끼친다

### V-Model



* V-모델의 테스트 레벨과 각 스테이지의 다섯 활동들은 (TMap 혹은 ISTQB) 애자일 테스트 방법의 일부이다.
* 초기의 하부 활동, 계획, 관리, 분석, 혹은 디자인은 테스트 방법, 방식, 측정 기준과 구조 등의 기본 절차의 준비에 포함된다.
* In Agile projects, this is done when the project starts (project initiation) or in Sprint 0. In particular, there is DoD criteria, such as Test coverage or number of errors allowed per User Story rather than per sprint or release.
* In the following activities test specification, implementation, and evaluation takes place for the components, component integration, and system test level within the Sprint. In contrast to a non-agile process, these tests are closely linked to the development of the application and are conducted in short iterative cycles, with a strong focus on automation of test execution.
* 마지막 단계는 스프린트의 각 끝과 출시 전에 일어나는 최종 공식 테스트 완료 단계이다.
* 스프린트 인터페이스가 비즈니스 프로세스를 통해 지속적으로 구현되지 않을 가능성이 있기 때문에, 큰 프로젝트들에겐 출시 전에 시스템 통합 테스팅을 위해 추가 시험 기간이 권장된다.

## 역할과 책임

### 팀 기반의 테스팅 방법

* 애자일 프로젝트 기간 동안 테스팅은 팀 중심이며 모든 팀 멤버는 일정 수준의 테스팅 지원을 제공한다.

### 테스팅에서 일반적인 제품 소유자들의 역할

* 일반적으로 제품 소유자들은 인수 기준에 지침을 제공하고 가끔 스토리들의 의도의 예를 제공하는 테스트 케이스를 만든다.

### 테스팅에서 일반적인 프로그레머의 역할

* 일반적으로 제품 소유자들은 인수 기준에 지침을 제공하고 가끔 스토리들의 의도의 예를 제공하는 테스트 케이스를 만든다.

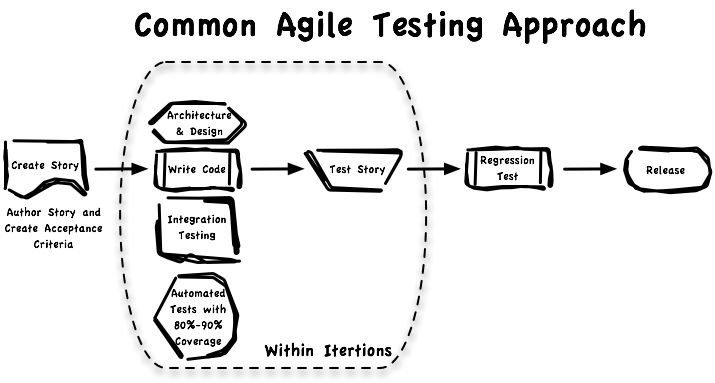
### 테스팅에서 일반적인 테스터의 역할

* Software programmers typically build, automate, and run a variety of tests at a variety of levels as part of their development process. TDD and ATDD leverages this testing to improve design and development.

### 테스팅에서 일반적인 고객의 역활

* Software testers typically work hand-in-hand with the product owner and programmers to plan, execute, and report on the testing that is performed at all levels. Testers often are responsible for creating User Story and business process tests cases and performing exploratory testing. Testers participate in and may develop automated tests along with programmers or a dedicated test automation team.

## 테스트 전략과 계획



### Light weight test plan documentation

* 애자일의 테스트 플래닝은 작업 수행에 필요한 문서를 최소화 시키는 게 목적이기 때문에 전통적인 개발 방법과 다르다.

### 스토리 추정

* 스토리의 크기는 애자일 팀이 지속 가능한 개발의 속도를 유지하면서 목표 속도를 달성하는 정보를 제공할 수 있도록 추정된다.

### 위험성-중심 테스팅 고려

* 테스트 케이스는 추가, 제거 및 요구 우선 순위에 따라 변경 될 수 있다

### 스마트 자동화 테스트 스크립트 생성

* 자동화 스크립트는 하나의 모놀리식 스크립트보단 재사용측면에서 생성한다. 자동화 테스트 스크립트는 각 스프린트마다 재검토 되기 때문에 스크립트를 간단하게 유지하는 것이 중요하다

### 테스트 활동들을 위한 플랜

* 테스트 스크립트를 다시 분사하기 위해서 시간 배분한다

테스트 관점을 유저 스토리 선발 과정에 포함한다

* 회귀를 위해 유닛뿐만 아니라 기능성 테스트를 다시 이용한다

유닛뿐만 아니라 기능성 테스트를 위해 최대한 테스트 케이스를 다시 사용하려고 노력한다

* 출시 전에 별도의 테스트 단계를 가진다

제품 테스팅 완성도 및 최적화를 보장하기 위해선 출시 전에 별도의 테스트 단계를 가지는 것이 최선의 방법이다

### 인수 기준

* Story acceptance criteria is critical to defining “done” for stories and provide guidance for the types of tests to develop during ATDD / story testing.

## 단위 중 테스팅

### 개발자/ 테스터 상호작용

* 애자일에선, 개발자와 테스터가 함께 스토리들을 빌드하고 테스트한다

### 테스터/ 고객 상호작용

* 애자일에선, 개발자와 고객은 인수 기준과 인수 테스트가 단위 마다 구현 및 테스트 될 스토리들에게 철저하고 적합할 수 있게 함께 작업합니다.

Defect tracking and management, the amount of defect tracking that is performed during an agile project depends upon what works best for the team.

### 결과 보고

* 애자일 프로젝트 기간 동안 테스트 보고는 팀에 가장 적합한 것에 따라 달라진다.

### 테스트 측정

* 테스트 완성도와 출시 준비 결정을 위해 측정한다

### 회귀 테스트

* 자동화 회귀 테스트는 변화 비용 감소와 개발 과정의 실시간 피드백을 제공에 있어서 필수적이다

### 단위 마무리

* 단위 기간 동안 마무리 활동은 제품 메모, 회고, 때로는 사용자 인수 테스트가 있다.

## 출시 중 테스트

### 출시/ 엔드 게임

* 출시 프로세스 (엔드 게임)는 소프트웨어를 고객에게 출시한다는 결정이 내려졌을 때 수행됩니다.

### 사용자 인수 테스팅 (UAT)

* 사용자 인수 테스팅은 출시 전 정상적으로 작동하는 소프트웨어에 고객의 피드백을 얻기 위해 애자일 내에서 사용된다.

### 스테이징 환경 테스팅

* 스테이징 환경은 생산과 유사한 환경 내에서의 어플리케이션의 테스팅을 지원하기 위해 설립된다

### 출시 후 테스팅

* 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

### 테스트에 대한 일반적인 환경

* 단위와 출시 프로세스 중에 테스트 활동을 지원하기 위해선 여러 환경이 필요하다

### 가상화

* 가상화는 테스트 과정 중에 효과적인 테스트 환경 설정, 테스트 실행 및 테스트 환경 해체를 지원하기 위한 매커니즘 (종종 자동화) 을 제공한다.

### 적절한 빌드 테스팅

* 빌드들은 지속적으로 애자일 프로세스에서 발생되므로, 적절한 빌드를 테스트하는 것은 효과적인 테스팅 과정에 중요하다

### 테스트 데이터 매니지먼트

* 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.

# 분산 된 팀에서의 작업

## ADM 내에서 분산된 애자일 테스트

* 이니시에이션: 통합, 성능 및 UAT에 대한 상위 레벨의 테스트
* 스프린트: 플래닝과 유닛, 희귀, 엔드 투 엔드 제품 테스트 실행
* 테스트: 통합, 성능, 엔드 투 엔드 테스트 및 UAT 테스트 실행

### 각 스프린트 내

* 컴포넌트 테스팅
* 어셈블리 테스팅
* 회귀 테스팅
* 어플리케이션 제품 테스팅

### 스프린트 벗어나서

* 통합 테스팅
* 성능 테스팅
* 사용자 인수 테스팅

# 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

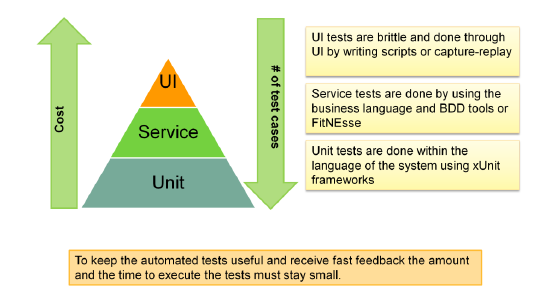


Figure 1. 마이크 코언(Mike Cohn)의 테스트 자동화 피라미드

* 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>