TBELL NEWSLETTER

Vol. 2 *2 April 2018*

1 JaSST'18 Tokyo 참가 Report

◆ 행사소개

3월 7일(수)~8일(목) 이틀간 동경 소재의 일본 대학에서 개최 된 JaSST(Japan Symposium on Software Testing)에 참석 하였다.

이틀 동안 총 35개의 세션이 진행 되었고 약 1600여명이 참석한 이번 행사는, 세계적인 추세를 방증 하듯 세션의 12개(35%)가 자동화와 관련된 것이었다. 청강을 하는 사람들 역시 자동화에 흥미가 많아 자동화 관련 세션장에는 자리가 부족할 정도였다. 일본 국내 행사이지만, 외국인 청강자 들도 심심치 않게 보여 이 행사에 가지는 해외의 관심도도 가늠할 수 있었다.

이번 TBELL NEWS에서는 행사의 가장 핵심이라고 할 수 있는 기조연설과 몇몇 세션에 대해 소개를 하고자 한다.

◆ 기조연설 : Advances in Continuous Integration Testing at Google

올해의 기조연설은 Google의 CI Testing & CD 팀의 수석 매니저인 John Micco가 강연자로 나섰다.

Google에서 실제 진행하고 있는 테스트 스케쥴링 방법과 테스트 비용 절감 방법, Flaky Test 대처법에 대해서 강연을 펼쳤다.

Google에는 **13,000개의 프로젝트 팀에 7만명 이상의 개발자들**이 일하고 있다. 이들에 의해 작성된 프로그램으로 **하루에 420만건의 테스트**가 실행되고 있으며 **1억5천만회의 테스트**가 실행되고 있다(한 건의 테스트에 35회 테스트 실행). 이 중 99 %가 테스트를 통과하고 **단 1%만이 테스트를 통과하지 못한다**고 한다. 이 1%의 오류를 잡기 위해 Google에서는 테스트를 실시하고 있는 것이다.

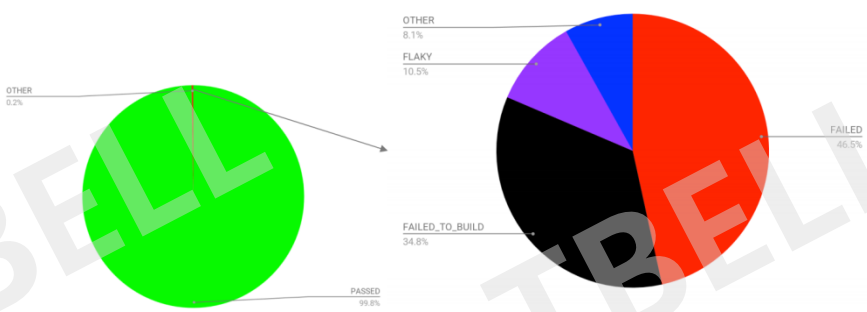
이 테스트는 **모두 자동화**를 통해 이루어지고 있다(외국어 번역과 관련된 된 부분 제외). 이것이 가능한 것은 Google에서는 사람의 생명과 관련된 중대한 사안을 다루고 있지 않다라는 배경도 있지만, 자동화 도입을 위해 오랜 시간 노력을 했기 때문이다. **2006년부터 자동화 테스트 도입**을 위해 회사의 시스템을 바꿨으며, GTAC

(Google Test Automation Conference) **개최**를 하고 있고, **2007년부터 테스팅 관련 블로그를 운영**하고 있다. 이런 노력으로 현재 Google에서 사용하고 있는 코드의 **97%가 자동화에 적합**하게 작성되어 있다. 그리고 나머지 3%도 언젠가는 수정을 통해 자동화에 적합하게 바뀔 거라고 한다.

Google은 프로젝트 별로 팀이 구성되어 있다. 팀의 인원은 프로젝트에 따라 다르지만, **8~10명의 개발자 마다 1~2명**의 테스트 인프라를 구축하는 업무가 주어진 **SETI 엔지니어**(Senior Tools & Automation Engineer)를 두고 있다. 그리고 결함 및 실수를 쉽게 찾을 수 있도록 **페어(pair) 프로그래밍**을 하도록 권장하고 있다.

개발자들은 아무리 짧은 코드라 할지라도 반드시 **코드가 변경되면 테스트 코드를 작성해 함께 제출**을 해야 한다는 룰을 따르고 있다. 그러면 이렇게 제출된 모든 프로그램에 의해 Google내의 모든 프로그램이 테스트 되는 것일까? 답은 그렇지 않다는 것이다. 리소스가 한정되어 있기 때문에 모든 코드를 테스트 하기 힘들며 그런 시간적인 여유조차 없다 하였다(개발자가 코드를 변경하면 36시간 내에 Web상에 배포하여야 한다는 룰이 있음). 이런 문제를 해결하기 위해 Google에서는 **상태의 천이**(Transition)에 주목하여 테스트 대상을 선별하고 있다. 하나의 변경에 의해 모든 것을 테스트 하는 것이 아닌 **의존 관계가 있는 부분만을 테스트** 하는 것이다.

Google에서는 모든 테스트 합격/불합격 결과에 태그를 붙여 2년간 저장하고 있다. 전략적 테스트 방법을 도출하기 위해 이 데이터를 분석한 결과 테스트의 99.8%에서 상태의 천이가 일어나지 않으며 **문제가 되는 것은 0.2%의 천이가 일어나는 부분**이라는 결론을 얻었다. 그렇다면 이 천이가 일어나는 부분에 대해서는 모두 테스트를 진행하고 있을까? 이 역시 그렇지 않다. 리소스 절약을 위해 꼭 테스트를 해야 하는 천이를 골라내어 테스트를 진행하고 있다. 이 중 하나가 Greenish Service이다.



[그림1] 데이터 분석 결과

여기서 Green이라고 하는 것은 합격을 의미한다. **100% 테스트에 합격 할 것이라고 판단된 것의 테스트는 실행하지 않는 것이다.** 각 팀 별로 리소스 사용에 대한 비용을 부담하게 하여, 팀에서 비용 절감을 위해 어느 정도 리스크를 가지고 릴리즈 하게 하는 것이다. 단 팀 별로 릴리즈의 책임이 있기 때문에 단순히 비용을 절감하려고만 하지는 않는다.

또 한가지는 Flaky Test에 대한 대처법이다. **Flaky(일관성 없음, 불안정함)라는 것은 같은 상태에 대해서 테스트 상황이 바뀜에 따라 결과가 변화 하는 상태**를 이야기 한다. 테스트의 16% 정도가 Flaky한 상태이며 이것은 **릴리즈를 방해하는 큰 요인**이 된다. 테스트의 **천이와 Flaky 사이에는 비례 관계**가 있어서 천이가 많은 테스트는 Flaky의 확률이 높아진다고 이야기 할 수 있다. 따라서 **천이가 얼마나 일어났는지를 찾음으로서 Flaky의 여부를 확인** 할 수 있다. Flaky를 일으키는 요인에는 3명 이상의 개발자가 코드에 관여하는 경우, 프로그래밍 언어, 실행 환경(chrome, Android), 테스트 케이스 펙터(UI 테스트, 웹드라이버 테스트 등) 등이 있으며, **어느 정도의 Flaky는 허용**해야 한다는 것을 인지해야 한다.

현재 Google에서도 시대의 흐름에 따라 **AI를 이용한 테스트**를 하고 있다. **AI를 통해 테스트를 선정하고 스케줄링**을 하고 있지만, 기계학습을 통한 테스트는 학습에 대한 신뢰성(수학적인 문제에 대한)이 떨어지기 때문에 하고 있지 않다. 하지만 앞으로는 **기계학습의 알고리즘을 적용한 테스트가 테스트 업계의 큰 과제**가 될 것임에 인지하고 있으며 적용을 위해 노력하고 있다.

◆ 초청강연 : 내가 경험한 소프트웨어 테스트의 변천

블록체인 기술 개발과 이와 관련된 어플리케이션과 서비스를 제공하는 소라미츠의 설계 컨설턴트인 시바타 요시키씨의 강연이었다. 자신이 대학원 시절 경험했던 프로그램 개발부터 후지제록스에서의 경험을 통해 앞으로 **어떤 자세로 소프트웨어 테스트에 임해야 하는지**에 대해 이야기 하였다.

시바타씨 역시 **계속적 통합과 자동화**를 강조하였다. 계속적 통합 및 자동테스트를 하는 것이 강점인 시대는 이미 지났으며, **이것들을 하지 않으면 도태되는 시기**가 되었음을 강조하였다. 과거에는 '작업'과 '창조적인 활동'을 모두 사람이 했지만

작업으로 생긴 시간의 딜레이가 '창조적 활동'에 영향을 주는 경우가 많기 때문에 **'작업'을 자동화로 함으로서 창조적 활동에 더욱 집중해야 살아 남을 수 있다**는 것이다.

강연을 통해 일본도 소프트웨어 업계의 상황도 우리나라와 크게 다르지 않다는 것을 알 수 있었다. 많은 사람들이 계속적 통합을 도입하고 싶어하지만 여러 가지 이유로 도입하지 못하고 있는 실정이었다. 아직도 **1990년대의 개발 수법만을 고수**하고 있는 곳이 많으며, **개발자들이 테스트 코드가 없는 소프트웨어만을 취급**한다는 점, 그리고 이러한 환경이 지속되어 **계속적 통합을 경험해 본 적이 없는 매니저들**이 대부분이고 그렇기 때문에 **후진양성을 할 수 없는 상황의 반복**이 계속적 통합이 도입되지 못하는 이유라고 하였다.

시바타씨는 고통이 따르겠지만 이러한 인식의 변화가 필요하다고 주장했다. 많은 기업들이 QA를 개발 프로세스의 가장 마지막이라고 생각하는 곳이 많으나, 계속적 통합을 위해서는 'Test First'를 도입하여 QA가 프로세스 전체를 통해 깊이 관여해야 함을 강조하였다. 단순히 소프트웨어를 만드는 것이 아니고 '좋은 품질'의 소프트웨어를 만들기 위해 **개발 초기 단계**에서부터 계속적 통합을 위해 관심을 가지고 **자동화가 되도록 추진**해야 하며, 자동화 된 각각의 테스트는 정확하게 **개발 부문에서 테스트 설계**가 되도록 협력해야 한다고 하였다.

이미 깊이 뿌리 박힌 인식의 변화를 위해 각 **기업별로 프로그래밍 '룰'을 책정**할 것을 제안하였다. 코드 작성시에는 배포 할 것만을 생각 하지 말고 **클래스와 모듈 단위로 명확한 API설계**를 한다든지, **버그를 수정할 때는 재현 테스트 관련 코드를 먼저 작성**하여 버그를 재현한 후 실제 배포 상황에서 수정을 한다 든지, **계속적 통합이 실패하면 최우선으로 수정**을 한다는 등의 규율을 통해 계속적 통합 및 자동화 도입이 가능해질 것이라고 하였다.

변화에는 많은 노력이 필요하다. 하지만 변화하지 않으면 살아남을 수 있는 시대가 되었음을 받아들이고, 시바타씨의 주장에 대해 깊이 생각해봐야 할 때라고 생각한다.

◆ 주요 세션

1. UI 자동화 툴과 AI

AI를 사용한 모바일 어플리케이션 테스트 자동화 Web서비스 'Magicpod' 개발사 TRIDENT사 대표의 강연이었다. 일본이 테스트에 사용하고 있는 비용은 연간 약 5조엔에 달하며, 그 중 많은 부분이 UI를 기반으로 하는 매뉴얼 테스트이기 때문에 이것을 자동화하여 효율을 극대화 하는 것이 목표라고 하였다. 현재 배포중인 프로

그램에 **딥러닝을 통한 학습기능을 추가**하여 아이콘 버튼을 시각적으로 인식하는 프로그램을 개발 중이다.

2. 고속 디버그 로그 분석과 자동화

게임으로 유명한 'Sega'의 테스트 담당자의 강연이었다. 게임 개발의 테스트 플레이와 툴 사용 중에 다양한 로그가 발생을 하는데, 이 때 발생하는 **디버그 로그**를 오픈소스인 **Fluentd + Elasticsearch + Kibana** 의 조합으로 데이터를 수집, 축적, 분석하여 자동 플레이 테스트를 실행한 구체적 사례에 대해 설명하였다. Jenkins와 Redmine등과 연계함으로써 개발과 테스트 코스트를 삭감하였고 이것이 게임의 품질과 재미 향상에 연결되었다는 내용이었다. 상용 툴이 아닌 **오픈 소스 툴을 사용**하고 있다는 점이 흥미로웠다.

3. 패널 디스커션

야후의 개발자인 야마구치씨를 통해 야후의 테스트에 대해 알 수 있었는데, Google과 같이 모든 프로그램을 자동화 하고 있지는 않지만 많은 부분에서 자동화를 도입하고 있다고 한다. 야후 역시 **테스트를 개발과 분리하지 않고** 개발하는 중에 테스트를 실행하도록 되어 있으며, 이를 통해 **개발자가 서비스 향상에 대한 인식이 높아져서 생산성이 높아졌다**고 한다.

기조연설을 했던 Micco씨는 일본 테스트 업계의 문제에 대해서 다음과 같이 말했다. **일본의 기업 및 임원들은 QA엔지니어의 임금을 낮추고 싶어하며, 개발자들에게 테스트 코드를 작성하게 하는 것이 생산성 저하와 연관이 된다고 생각하고 있다**는 것이다. 그는 기업 및 임원들이 테스트 **자동화를 통해 얻을 수 있는 더 큰 가치를 깨달아야 한다**고 강조하였다.

여러 강연을 들으면서 끊임없이 **자동화 도입의 중요성**에 대해서 확인할 수 있었다. 자동화 도입은 **선택이 아닌 필수**가 되었다. 조사에 따르면 2019년까지 테스트 자동화를 도입하지 못한 프로바이더는 살아 남지 못할 것이며, 2020년에는 50%의 기업이 OSS와 프레임 워크를 사용하여 계속적 통합을 하게 될 것이라고 한다.

지금 생존을 위해 자동화 도입을 해야 할 때이다!

2 애자일 테스트와 지속적 전달(Continuous delivery)로 모바일 DevOps 개선하기

본 내용은 Test Magazine 3월호에서 발췌/번역한 기사로 작성자는 Mobile Labs CEO의 DAN MCFALL입니다.

구현할 때 중요한 것은 프로세스를 잘 정의하고 단순하게 유지하며, 개발, 테스트 그리고 품질보증이라는 목적을 달성하기 위해 적절한 툴을 제공하는 것이다.

엔터프라이즈 모빌리티는 한 순간도 멈춰 있지 않다. 디지털 변환으로 응용 프로그램 개발 및 테스트 팀에게 새로운 과제를 제공하는 모바일 장치 및 운영 체제가 지속적으로 나오고 있다. 엔터프라이즈 모빌리티 팀은 가용 리소스와 비용에 맞춰 정해진 시간에 요구사항을 충족하는 고품질 모바일 앱과 모바일 웹사이트를 제공하기 위해 노력하고 있다.

하지만 엔터프라이즈 모빌리티 팀은 끊임 없는 혁신으로 야기되는 문제를 어떻게 해결 하였을까? 정답은 **모바일 DevOps 프로세스를 개선**하는 것이었다.

모바일 DevOps의 현재

DevOps는 새로운 개념이 아니지만 많은 엔터프라이즈 모빌리티 팀은 DevOps의 접근 방식을 완벽하게 구현하기 위해서 지금도 노력하고 있다. 더 많은 모바일 개발자, 테스터 및 품질보증 전문가가 이전 보다 더 빠르게 좋은 앱과 모바일 웹사이트를 개발하고 테스트하여 릴리즈해야 한다는 압박을 계속 받고 있기 때문에, **프로세스를 간소화하여 더욱 빠르고 효율적으로 만드는 것이 중요**하다.

모바일 DevOps의 골과 이니셔티브의 달성은 **애자일 테스트**와 **계속적 전달**을 이라는 두 가지 방법을 통해서 가능하다. 계속적 전달을 활용함으로써 모바일 테스트에 있어 테스터 및 품질보증 팀의 구성원들이 서로 잘 **연계** 할 수 있게 하여 **적시에 적절한 작업을 수행 가능**하게 한다. 개별적으로 봤을 때, 모바일 테스트 자동화만으로 애자일 테스트를 하기에는 부족함이 있다. 모든 자동화된 테스트는 지속적 전달 툴을 통해 액세스 가능하고 실행 할 수 있어야 한다. **모바일 DevOps를 만드는 것은 지속적 전달 프로세스**라 할 수 있다.

애자일 테스트를 위한 자동화 테스트

애자일 테스트의 개념은 모바일 테스터가 앱과 모바일 웹 사이트를 빠르고 효율적으로 테스트 할 수 있게 하는 전반적인 '애자일 메니페스토' 와 일치한다. 애자일 개발 테스트 팀의 권한과 같은 것이 모바일 앱의 소프트웨어를 구축, 테스트 및 릴리즈 하거나 하는 경우에 자주 요구된다. 그러나 자주 릴리즈 하면 민첩함이 떨어진다.

테스트와 릴리즈를 통해 개발된 전체 프로세스는 반복적으로 실행가능 한 직접적인 프로세스여야 한다. **프로세스가 간소화되고 응집력있는 방식으로 실행되어야 개발, 테스트 및 제공시간이 지속적으로 빨라질 수 있**기 때문이다.

자동화 테스트가 릴리즈가 많은 모바일 앱과 모바일 웹 사이트를 테스트하기 위한 이상적인 솔루션임에 틀림 없지만 테스트 자동화를 통합할 때에는 다음과 같은 사항을 고려해야 한다.

테스트 자동화 전략을 세울 때 구현 전에 **필요한 테스트 유형을 고려**해야 한다. 그리고 **유닛, 기능, and/or 퍼포먼스에 관계없이 테스트 형식 및 시험의 양을 고려**하는 것도 중요하다. 테스트 해야하는 앱과 운영 체제 및 디바이스의 수 등을 고려하여 소비자에게 릴리즈 하기 전에 **대량 테스트를 의무적으로 시행**해야 한다.

애자일 테스트를 최대한 활용하려면 **올바른 테스트 툴을 선택**해야 한다. 전체 팀의 역량을 가장 잘 보완하고 활용하는 툴을 사용하면 모든 구성원의 **생산성을 높일 수 있으며**, 특히 공동 작업과 생산성을 향상시키는 솔루션을 활용하면 더욱더 생산성이 향상된다. Appium과 같은 오프소스 테스트 프레임 워크를 선택하거나 MicroFocus의 UFT와 같은 상용화 툴을 선택하여 적절한 도구를 테스터가 사용하게 함으로써 성공적으로 테스트를 진행하여 릴리즈 시기를 놓치지 않게 할 수 있다. 하지만 자동화 테스트는 모바일 DevOps를 향상시키는 지속적 전달의 파이프라인을 성공적으로 구축하는 수단의 하나일 뿐이라는걸 알고 있어야 한다.

계속적 전달을 통한 강력한 모바일 DevOps

계속적 전달을 위한 파이프라인을 구축할 때 팀의 **모든 구성원들은 역할을 분담**해야 한다. 개발자, 테스터 및 품질보증 전문가 모두가 빌드와 모바일 앱 릴리즈의 다양한 측면에서 책임을 지고 있지만 계속적 전달 프로세스 구축은 모든 단계에서 **협력 및 피드백이 가능**해야 한다.

구현할 때 **프로세스를 명확하게 정의하고 단순하게 유지**하는 것이 중요하다. 개발, 테스터 및 품질 보증을 위해 적절한 툴을 제공해야 하는데, 릴리즈 후 모니터링 도구를 사용하여 버그를 빠르게 수정할 수 있도록 동료 및 팀원에게 이슈와 피드백 할 수 있기 때문이다.

협력이 가능하면 혁신 및 모바일 DevOps는 올바른 성장을 하여 좋은 결과를 보여줄 수 있다.

미래 혁신을 위한 준비

애자일 테스팅과 계속적 전달을 모바일 DevOps 이니셔티브의 일환으로 구축함으로써 기업은 앞으로의 상황(미래)에 대비 할 수 있다. 앞으로 어떤 새로운 모바일 트렌드와 기술이 등장해도 직면한 문제를 효과적으로 해결할 수 있을 것이다. **간소화된 DevOps 프로세스를 구축함으로써 모바일 개발자, 테스터 및 품질 보증 팀은 변화와 혁신을 고려하여 모바일 DevOps를 계속 개선할 수 있는 기반을 마련할 수 있을 것**이다.