RAKTA: 키워드 기반 탐색적 테스팅 자동화

황준선*, 최은만*
*동국대학교 컴퓨터공학과
e-mail: hwangsun12@naver.com

RAKTA: Automation of Exploratory Testing Based on Keyword

Jun-Sun Hwang*, Eun Man Choi*
*Dept of Computer Science, Dongguk University

요 약

일반적인 키워드 기반 테스트는 기능 위주의 키워드를 작성하여 테스트를 자동화하여 비용은 적게 들지만 활용도가 높은 테스트를 자동화기 어렵다. 한편 탐색적 테스트는 리스크 기반으로 차터를 작성하여 짧은 시간동안 많은 에러를 탐지하는 장점이 있으나, 문서화가 미흡하다는 단점이 있다. 위와 같은 단점을 보완하기 위하여 탐색적 테스트의 기본 원리를 고수하면서 효율적 키워드 기반 자동화가 가능한 RAKTA(Record And Keyword-based Test Automation) 방법론을 제안한다. RAKTA는 오픈 소스 키워드 기반 자동화 프레임워크인 로봇 프레임워크의 기술을 사용하여, 키워드 기반과, 탐색적 테스트의 장점을 뽑아 효율적으로 테스트 자동화하여 비용을 줄이고 많은 에러를 탐지할 수 있다. 또한 본 논문에서는 RAKTA 방법론을 활용한 여러 가지 키워드 재사용 사례와 기존 조직에서 사용하던 테스트 스크립트를 혼합하여 통합 테스트, 인수 테스트, 설치 테스트를 자동화하는 방법을 제안한다.

1. 서론

소프트웨어 테스트의 목적은 소프트웨어의 에러를 탐지하고 품질을 높이는 것이다[1]. 키워드 기반 테스트는 주로 기능 위주로 작성이 되어있다. 기능 테스트 키워드들은 기본적인 테스트 커버리지를 달성하는 데는 도움이 되나, 사용자들의 독특한, 예상하지 못한 동작들을 탐지하기에는 어려움이 있다. 본 논문의 연구는 다음과 같은 연구 물음에 답을 찾으려는 것이다.

연구 질문 1. 리스크가 큰 에러를 효율적으로 탐지하는 키워드를 만들 수 없을까?

규모가 큰 프로그램을 운영할 때 단위 테스트는 개발자들이 직접 테스트 스크립트를 직접 작성하여 QA팀에게 전달한다[2]. 개발자들이 이러한 테스트 스크립트 작성에 많은 시간을 들이게 되어 비용이 많이 발생한다. 따라서 적은 비용으로 리스크가 큰 오류를 효율적으로 탐지하는 키워드를 만드는 방법을 연구하려 한다.

연구 질문 2. 키워드 자동화 비용을 낮출 수 없을까?

연구 질문 1과 2에 대해 탐색적 테스트 접근법으로 키워드를 작성하고 자동화하는 접근법을 통해 해결하고자 했다. 탐색적 테스트는 경험적 테스트에 리스크 기반 테스트 차터 작성, Time-Boxing, 경험 공유를 하는 접근법이다[3]. 탐색적 테스트는 효과적으로 오류를 탐지하기 위해 문서화에 비중을 적게 둔다. 즉 문서화가 미흡하고, 일반적으로 재사용 가능한 스크립트를 만들지 않는다는 단점이 있다.

연구 질문 3. 탐색적 테스트를 추구하면서 효율적 문서화가 가능할까?

탐색적 테스트를 고수하면서 효율적으로 문서화하기 위해 Record-and-Replay 기반으로 키워드를 만들고, 재사용하는 방법을 제안한다. 즉, 키워드기반 테스트와 탐색적 테스트가 서로 상호 보완하여 연구 질문들을 해결한다.

제안하는 방법은 오픈소스 키워드 기반 테스트 프레임워크인 로봇 프레임워크의 기술을 사용하여 설계한다. 로봇 프레임워크에 테스트 녹화 기능을 추가하여, 테스트를 녹화하면 쉽게 테스트 키워드를 만들 수 있고, 로봇프레임워크에서 제공하는 라이브러리와, 조직에서 기존에 운영하던 테스트 스크립트, 동작 녹화 키워드를 혼합하여 사용하여 확인 테스트, 회귀 테스트, 회고, 통합 테스트, 시스템 테스트, 인수 테스트, 설치 테스트에서 재사용하는 방안을 제안한다.

2. 연구 배경

일반적으로 자동화 테스트의 문제는 테스트 활용도에 비해 자동화 비용이 높고 오류탐지와, 재사용성을 비교하였을 때 효율이 낮다[5]. 또한 새로운 자동화 프로세스를 도입하였을 때 기존 조직에서 사용하던 테스트 케이스를 통합할 수 없어 비용이 많이 든다. 키워드 자동화 비용을 낮추기 위한 도구로 테스트 상용풀인 Telerik Test Studio가 있다. 포인트 앤 클릭으로 자동화를 간단하게 해주고, 코드 분석을 할 필요가 없어 유지보수 비용을 줄여준다.[8] 그러나 포인트 앤 클릭을 제외한 다른 키워드를 혼합하여 사용하는 기능이 없고. 기존 테스트 케이스를 재사용 할 수 없다. kUTAF라는 유사한 자동화 도구가 있는데 이는 도메인 지정 언어를 사용하여, 도메인 내 구성원이면 누구든지 테스트 케이스를 이해할 수 있어, 자동화

효율성을 향상시킨다[7]. 그러나 Telerik Test Studio와 같이 재사용 단점이 존재한다.

로봇프레임 워크는 키워드 기반 테스트 자동화를 지원하는 오픈소스 프레임워크이다. 오픈 소스로 다양한 라이브러리들을 제공하여, Telerik Test Studio와 kUTAF이 할 수 없던 여러 가지 키워드를 통합할 수 있다. 키워드 또한 도메인 지정 언어로 설정할 수 있고, 광범위한 라이브러리를 추가 및 확장하여 자동화할 수 있다.

3. 자동화 효율성 향상 방안

키워드 기반 테스트들과 다른 접근 방법으로 여러 종류의 키워드들을 혼합하여 테스트를 자동화 하는 방안에 대해 소개한다. 여러 종류의 키워드를 혼합하여 테스트 케이스를 작성할 때 오픈소스 키워드기반 자동화 프레임워크인 로봇 프레임워크를 사용한다.

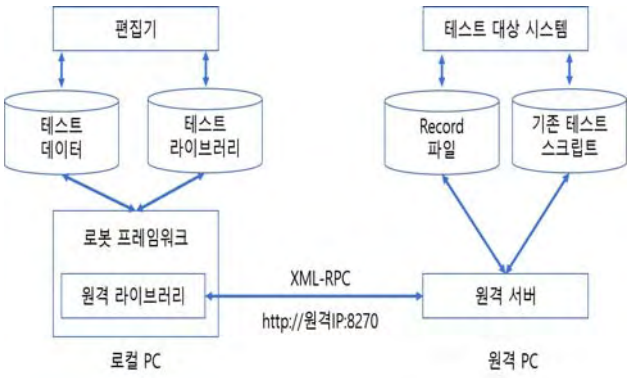

그림 1 RAKTA(Record And Keyword based Test Automation) 개념도

그림 1은 로봇프레임워크를 사용하여 제안하는 방안인 Record And Keyword based Test Automation이다. 제안하는 방안인 RAKTA는 “키워드 자동화 비용을 낮출 수 없을까?” 라는 연구 질문을 해결하기 위해 설계하였다. 기존 키워드 기반 테스트 자동화에 Record를 사용하여 동작을 녹화 할 수 있는 기능을 추가하였다. 녹화한 동작은 키워드로 재사용이 가능하다. 녹화한 동작을 키워드로 만드는 과정은 아래와 같다.

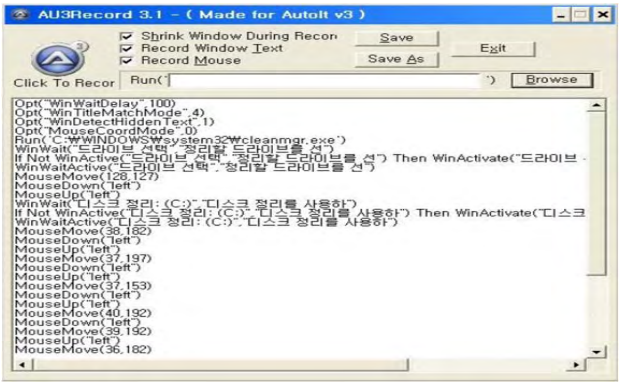

그림 2 AU3Recorder 동작 녹화 프로그램 실행

1. 모바일은 “RERAN”, PC는 그림 2와 같이 “AU3Record” 프로그램을 사용하여 동작을 녹화한다.
2. 그림 3과 같이 로봇프레임워크에서 제공하는 라이브러리를 사용하여 동작이 녹화된 파일을 실행시킨다.
3. “Pass”, “Fail” 결정은 라이브러리를 사용하여 작성한다.

```
Run_Record_Test
${test}=            Set Variable      Login.exe
${test_summary}=          Set Variable      앱 로그인

AI.run              ${test}
AI.Win Exist        ${test}
Log To Console              ${test_summary}
Exit For Loop If            AI.Win Exist      ${test}
Take Screenshot             ${test}           800px
```

그림 3 녹화 파일 실행 키워드 예시

제안하는 방안에서 사용하는 키워드의 종류는 “테스트 라이브러리”, “녹화 키워드”, “스크립트 키워드” 등 세 가지가 있다. “테스트 라이브러리”는 로봇 프레임워크에서 제공하는 라이브러리로 “Selenium“, “Appium“, ”Android“, “Autoit“ 등 여러 가지 환경에서 사용할 수 있는 키워드를 제공하여 준다. 이 라이브러리는 로봇 프레임워크가 동작하는 PC에서 XML-RPC통신을 사용하여 테스트 대상을 동작시킨다. ”녹화 키워드“는 ”테스트 라이브러리“로 만든다. 모바일 또는 PC에서 매크로 툴을 사용하여 동작을 녹화하면 녹화 파일이 생성된다. 이 녹화된 파일을 그림 3과 같이 ”테스트 라이브러리“를 사용하여 원격으로 실행시키고, 수행 후 ”Pass“, ”Fail“ 조건은 테스터가 결정한다. ”스크립트 키워드“는 조직에서 기존에 사용하던 테스트 케이스들이다. 기존에 사용하던 테스트들을 모듈별로 새로운 키워드를 지정하여 주면, 로봇 프레임워크의 Remote 라이브러리를 사용하여 원격으로 기존 테스트 케이스를 재사용 할 수 있다. 세 가지 키워드 모두를 사용하여 로봇프레임워크 환경에서 테스트 케이스를 작성할 수 있다.

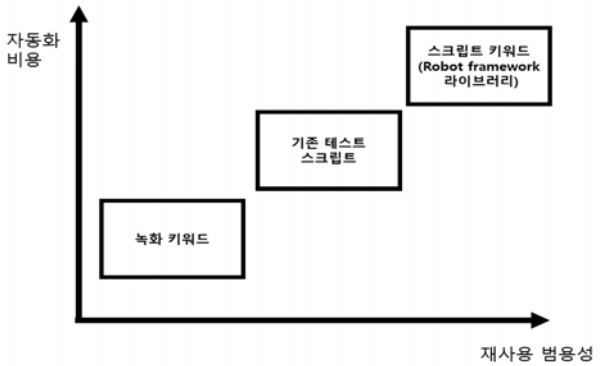

그림 4 키워드별 자동화 비용과 재사용 범용성

그림 4은 RAKTA에서 사용하는 세 가지 키워드의 자동화 비용과 테스트 범용성에 대한 비교표이다. 여기에서 테스트 범용성이란, 테스트 라이브러리를 사용하여 작성한 테스트의 경우에는 한번 작성하면 유지, 보수가 쉽고 또한 여러 가지 기기들을 동시에 테스트 할 수 있다. 그 대신 키워드 작성하는데 비교적 많은 시간을 필요로 한다. 예를 들어 로봇 프레임워크의 Appium 라이브러리를 사용하여 테스트 케이스를 작성하게 되면 IOS, 안드로이드 플랫폼 모두에서 애플리케이션을 테스트 할 수 있다. 또한 UI가 변경되거나, 화면 사이즈가 다르더라도 모두 테스트 할 수

있다. 반면에 녹화 키워드는 녹화한 동작을 분리할 수 없고, 화면 사이즈가 다르거나, UI가 변경되면 재사용하기 어렵다, 그 대신 수동 테스트와 비용이 차이가 없을 정도로 자동화 비용이 적게 든다. 조직에서 사용하던 기존 테스트 스크립트의 경우에는 자동화 비용도 적고, 재사용 범용성도 적당하여 필수적으로 재사용하는 것이 좋다.

그림 5 테스트 자동화 전략

그림 5는 상황에 따른 테스트 자동화 전략을 나타낸 것이다. RAKTA를 사용하면 기존에 키워드 기반 테스트 자동화에서는 "수동 테스트", "가능한 자동화"에 있는 것을 "자동화 고려"와, "반드시 자동화"로 자동화 비용을 낮출 수 있다. 예를 들어 복잡한 동작이 들어가 스크립트를 작성하는데 비용이 많이 들면, Record 기능을 사용하여 동작을 녹화하여 자동화 한다. 테스트 활용도가 높고, 도입 비용이 낮은 경우인 "반드시 자동화"에 범주에 들어가는 테스트는 라이브러리를 이용하여 테스트 스크립트를 작성하여, 유지, 보수와 범용성이 넓게 해준다. 기존 테스트 스크립트의 경우에는 테스트 활용도가 높으면서, 도입 비용이 낮으므로 반드시 자동화 해준다. 모든 자동화된 키워드는 단순히 에러 탐지에만 사용되지 않고 탐색적 테스트의 전과정에서 범용적으로 사용된다.

4. 탐색적 테스트에서 활용 사례

이번 장에서는 "리스크가 큰 에러를 효율적으로 탐지하는 키워드를 만들 수 없을까?"와 "탐색적 테스트를 추구하면서 효율적인 문서화를 할 수 없을까" 라는 두 가지 연구 질문에 대한 해결책을 제시한다.

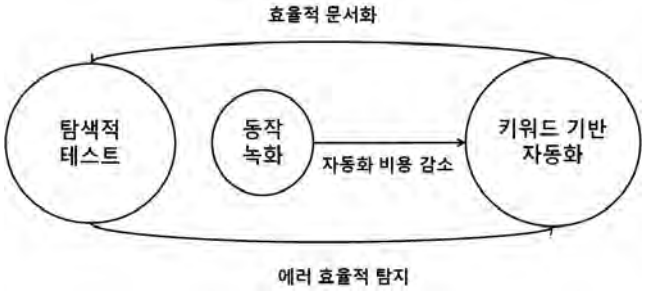

그림 6 탐색적 테스팅에서의 RAKTA 접근법

그림 6은 탐색적 테스팅에서의 RAKTA 접근법이다. RAKTA에서 제공하는 동작 녹화기능을 사용하여 탐색적 테스팅에서 효율적인 문서화를 한다. 오류 탐지도 테스트의 중요한 목적 중 한가지인데, 키워드 기반 테스트는 키워드 작성을 기능 테스트 위주로 한다.

표 1 리스크 분석 테이블

리스크 요소 / 리스크 아이템	장애 발생 가능성 (Likelihood)			장애로 인한 영향도 (Impact)		
리스크 아이템 1						
리스크 아이템 2						
. . .						
리스크 아이템 n						

리스크가 큰 에러를 효율적으로 탐지하는 키워드를 만들기 위하여 표 1과 같이, 리스크 분석 테이블을 사용한다. 리스크 레벨 선정은 9=심각, 5=높음, 3=보통, 1=낮음, 0=없음으로 영향도를 구분하여 우선순위를 결정한다. 탐색적 테스트에서 리스크 분석 테이블을 기반하여 테스트 차터를 작성한다. 테스트 차터에는 리스크가 높은 아이템에 더 많은 테스트 시간을 할당한다. 그 다음 차터를 기반으로 정해진 Time-Boxing동안 몰입해서 탐색적으로 오류를 찾는다. 테스터가 발견한 오류는 RAKTA의 녹화기능을 사용하여 Record 파일로 만들어준다. Record 파일의 이름은 탐지한 에러에 대한 간단한 키워드로 작성해준다. 위 과정을 통해 탐색적 테스트에서 RAKTA를 사용하여 효율적 문서화와, 에러를 잘 탐지하는 키워드를 만들 수 있게 된다.

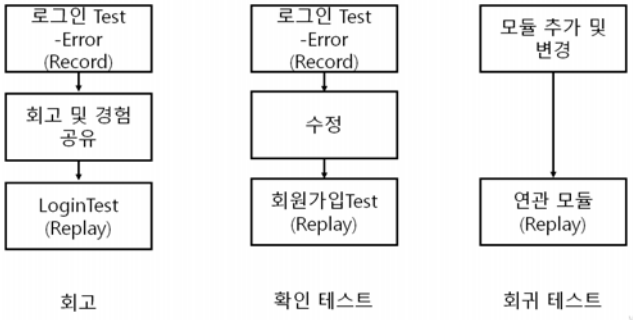

그림 7 RAKTA를 사용한 키워드 재사용

탐색적 테스트에서 만든 키워드를 RAKTA를 사용하여 재사용하는 사례는 대표적으로 세 가지가 있다, 회고에서의 재사용, 확인에서의 재사용, 회귀에서의 재사용으로 활용 방법은 그림 7과 같다. Time-Box가 끝나게 되면 탐색적 테스트에서는 회고를 통해 어떤 에러가 있는지 경험을 공유를 하게 된다. 이때 녹화한 Record 파일을 수행하여 실제 동작을 보여주며, 어떤 에러가 발생하는지 효과적으로 이해하고, 경험을 공유할 수 있다. 이 키워드들은 따로 회고를 하지 않고 개인이 따로 녹화 파일만 전달 받아도 경험을 공유할 수 있게 된다. 또한 키워는 확인 테스트에서도 재사용할 수 있다. 개발자가 오류를 수정하고 나서 오류가 잘 수정되었는지 확인 테스트를 통하여 확인한다. 다음은 회귀 테스트에의 재사용이다. 모듈을 변경하거나 추가할 때 부작용이 없는지 이전에 발생했던 에러가 재발생하지 않는지 키워드를 수행하여 테스트 한다. 탐색적 테스트에서 만든 키워드는 테스트 라이브러리와, 기존 테스트 스크립트, Record 키워드와 함께 사용하여 통합테스트, 시스템테스트, 인수테스트, 설치테스트에서 가능하다. 통합

테스트는 모듈끼리 연결을 하였을 때 문제가 없는지 확인할 때 사용한다. 그림 8에 대한 통합 테스트 케이스 예시는 아래 그림 9와 같다.

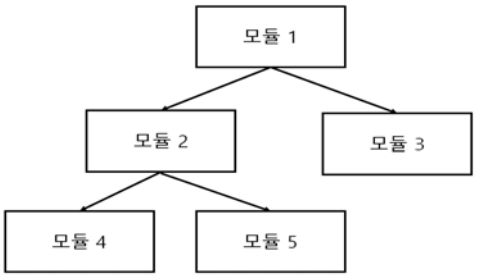

그림 8 테스트 대상 예시

통합 Test case 1

1. 모듈 1 Test (Replay & Scripts)
2. Click 모듈 2 (Android Libray)
3. 모듈 2 Test (Replay & Scripts)
4. Click 모듈 4 (Android Libray)
5. 모듈 4 Test (Replay & Scripts)

통합 Test case 2

1. 모듈 1 Test (Replay & Scripts)
2. Click 모듈 2 (Android Libray)
3. 모듈 2 Test (Replay & Scripts)
4. Click 모듈 5 (Android Libray)
5. 모듈 5 Test (Replay & Scripts)

통합 Test case 3

1. 모듈 1 Test (Replay & Scripts)
2. Click 모듈 3 (Android Libray)
3. 모듈 3 Test (Replay & Scripts)

그림 9 통합 테스트 케이스 작성 예시

모듈 1을 테스트할 때, 탐색을 통해 만들어 놓은 Replay와 기존 조직에서 사용하던 스크립트를 동작시켜 테스트한다. 다음 모듈로 넘어가는 것은 로봇프레임워크에서 지원해주는 라이브러리인 스크립트를 이용하여 작성한다. 다음 모듈에서 또 이전에 탐색을 통해 녹화한 키워드와, 기존 스크립트를 동작시킨다. 이를 통해 통합테스트에서의 재사용이 가능하다. 설치 테스트는 이러한 통합테스트 키워드를 슈트로 통합하고 설치 동작을 녹화하고, 설치가 끝난 후에 모든 동작이 오류 없이 동작하는지 확인할 수 있다. 인수 테스트 또한 고객에게 이러한 동작이 잘 수행되는 것을 직접 빠르게 보여주어 이해를 돕는다.

5. 결론 및 향후 연구

본 논문은 비효율적인 테스트 자동화에 대해 3가지 연구 질문을 던지고 이를 해결하는 방안을 제안 하였다.

“연구 질문 1. 리스크가 큰 에러를 효율적으로 탐지하는 키워드를 만들 수 없을까?“

연구 질문 1을 답하기 위해 제안하는 방식인 RAKTA를 사용하여 키워드 기반 테스트와, 탐색적 테스트의 접근을 혼합하여, 리스크 기반 테스트 차터를 사용하여 리스크가 큰 에러를 효율적으로 탐지하는 키워드를 만드는 방안에 대하여 제안하였다.

“연구 질문 2. 키워드 자동화 비용을 낮출 수 없을까?”

연구 질문 2에 대한 해결책으로 회고, 확인, 회귀의 테스트를 Record 키워드로 작성하여 낮은 비용으로 재사용여 자동화 비용을 효과적으로 줄이고, 또 Record 키워드, 라이브러리 키워드, 기존 테스트 스크립트를 혼합하여 테스트 케이스를 작성하여 테스트 자동화 비용을 낮추고, 유지, 보수를 효율적으로 할 수 있는 방안을 제안하였다.

연구 질문 3. 탐색적 테스트를 추구하면서 효율적 문서화가 가능할까?

연구 질문 3에 대한 해결책으로 탐색을 할 때 문서화를 RAKTA에서 제공하는 Record기능을 사용하여 키워드로 만들어 주어 문서화한다. 이 방법을 통해 탐색적 테스트를 방해하지 않으며 효율적인 문서화가 가능하게 된다. 또 이렇게 만들어진 키워드는 연구 질문 1과 2에 대한 해결책이 된다.

향후 연구로는 현재 RAKTA를 실제로 구현하여 탐색적 테스트에 방해가 되지 않게 문서화를 할 수 있도록 편리하게 제작할 예정이다. 또한 이 방법이 실제로 현업에서 효과가 있는지 실제 실험을 통한 검증이 필요하다. 또한 본 논문에서 제안한 자동화 사례 외에도 여러 가지 효율적 자동화가 가능한 부분이 있는지 추가적으로 연구할 필요성이 있다.

참고문헌

[1] 권원일 외, “개발자도 알아야 할 소프트웨어 테스팅 실무, 제3판”, ㈜STA테스팅컨설팅, 2010
[2] 권원일, “SW Quality Insight 제 35회 SW 공학 테크니컬 세미나 탐색적 테스팅(Exploratory Testing)”, https://youtu.be/uO8wg8bVeRI, (주)STA테스팅컨설팅 , 2013
[3] Elisabeth Hendrickson, “EXPLORE IT REDUCE RISK AND INCREASE CONFIDENCE WITH EXPLORATORY TESTING”, Insight,2014
[4] STA테스팅컨설팅, “소프트웨어 테스트 실무 가이드”, ㈜STA테스팅컨설팅, 2012
[5] 정상미,”더 괜찮은 QA가 되기위한 프랙티컬 테스트 자동화”, 프리렉, 2018
[6] 조일상, “An Exploratory Testing Approach for Effective Defect Detection in Regression Testing : Focused on E-Commerce Services”, 강원대학교 석사학위 논문,2019
[7] 오정인, 김승아, 황정우, “Using Specification By Example and Keyword-based Test Automation for Agile Test”, KCSE, 2013
[8] 황영석, 정상문, 화창득, “A Keyword-based UI Test Framework for Web Service”, 정보과학회논문지: 소프트웨어 및 응용 제38권 제12호,2011
[9] James Bach, “Exploratory Testing Explained”, http://www.satisfice.com/articles/et-article.pdf, 2002
[10] ISO/IEC/IEEE DIS 29119-5 : Systems and Software Engineering - Software Testing - Part 5 : Keyword based Testing, 2013
[11] ISTQB, "Certified Tester foundation Level Syllabus", 2018
[12] N. Masato, C. Yoseke, "Cross-Platform application Development, Freelec, 2017
[13] s. Stresnjak, Z. Hocenski, "Usage of Robot Framework in Automation of Functional Test Regression", ICSEA, PP 30-34, 2011
[14] Robot Framework, "http://robotframework.org"