Record-Playback 기술 기반의 GUI 테스트 케이스 자동 생성

이정규° 김현수 국승학 조대완
충남대학교 전기정보통신공학부 컴퓨터전공
jjangqwer@nate.com, {hskim401, triple888. oopmania}@cnu.ac.kr

Record-Playback based Automatic test case generation for GUI test

JungGyuew Lee°, HyeonSoo Kim, SeunGhak Kuk, DaeWan Cho
Dept. of Computer Sc. & Eng., Chungnam National University

요 약

오늘날 GUI가 소프트웨어의 성공에 중요한 영향을 미침으로써 GUI에 대한 테스트가 요구된다. 반면에 소프트웨어의 GUI 테스트는 많은 노력과 시간이 소비된다. 이러한 자원의 소비를 줄이기 위해 GUI 테스트를 자동으로 수행하는 것이 필요하다. 본 논문에서는 GUI 자동 테스트를 수행하기 위해 극복해야 할 문제와 GUI 자동 테스트에 적용되는 기술에 대해 논하고, Record-Playback 기술을 이용하여 GUI 테스트 케이스 자동 생성 방법을 제시한다.

1. 서 론

오늘날 GUI(Graphical user interfaces)는 소프트웨어의 상호작용에 대한 중요한 접근 방법이 되었다. 비록 사용자 관점에서 GUI가 소프트웨어를 쉽게 사용할 수 있게 하지만, GUI는 소프트웨어 개발 과정을 복잡하게 만든다[1]. 또한 GUI 구현은 많은 소스 코드가 사용된다[2]. 특히 GUI 테스트는 기존의 소프트웨어 테스트 보다 더 복잡하다. 왜냐하면 GUI 테스트는 GUI의 작동에 대한 테스트와 소프트웨어 내부 동작에 대한 테스트가 병행되기 때문이다. 심지어 GUI 테스트 자동 도구는 소프트웨어의 오작동을 유발하는 GUI 에러를 발생시킬 수 있다[1].

최근에 Safety-critical 시스템에서 GUI 이용이 증가하고 있다. 이러한 시스템의 안정성, 견고성, 사용가능성을 증가시키기 위해 GUI 테스트가 수행되어야 한다[2].

정확한 GUI 테스트를 수행하기 위해선 몇 가지 어려움이 따른다[1].

첫 째로 GUI 테스트 범위는 다른 테스트 수행 범위 보다 양적인 면에서 많다. 서로 다른 GUI의 명령 조합은 다양한 많은 결과를 생성하고, 이 모든 결과는 테스트 되어야 한다.

두 번째로 기존 소프트웨어 테스트는 코드의 형태와 사용량을 측정하지만 GUI 테스트에서는 이러한 방법이 잘 적용되지 않는다. 왜냐하면 “얼마나 많은 코드가 테스트 되어지느냐?” 도 중요하지만 “얼마나 많은 GUI의 명령 조합들을 테스트 하는지” 역시 중요한 문제이기 때문이다. 그리고 테스트 케이스 실행 단계가 정확히 수행되는지에 대한 검증이 필요하다. 명확하지 않는 GUI에 대한 사용자 이벤트는 예상하지 못한 오류를 유발한다. 이러한 이벤트에 의한 테스트 케이스 생성은 부정확한 테스트를 수행하거나 오류 발생지의 검출을 어렵게 만든다.

세 번째로 GUI에 대한 회귀 테스트이다. 전형적으로 GUI 개발은 빠르게 변하는 프로토타이핑 모델을 사용하고 소프트웨어의 계속되는 버전 업그레이드는 지속적인 GUI의 변경을 초래하게 된다. 따라서 GUI 회귀 테스트는 특히 중요하다

GUI 테스트에서 GUI 테스트 케이스 생성은 중요한 구성요소이다. 하지만 앞에서 제시한 세 가지 문제점은 GUI 테스트 케이스를 생성하는데 어려움을 준다. 수동적으로 테스트 케이스를 생성/유지/평가하는데 많은 시간이 소비되고, 또한 테스트 범위 유지도 많은 시간이 소비된다[1]. 이러한 이유로 인해 GUI 자동 테스트가 필요하다.

본 논문에서는 Record-Playback기술을 이용한 GUI 테스트 케이스 자동 생성에서 이벤트 기록의 효율성과 활용성에 대한 기법을 제시한다. 본 논문은 2장에서 현재 적용되는 GUI 자동 테스트 기술을 소개하고, 3장에서는 Record-Playback기술에 대해 심층적인 접근을 기술하고 Record-Playback 기술을 이용한 GUI 테스트 케이스 생성 방법을 제시한다. 마지막으로 4장에서는 본 논문의 결론과 향후 계획을 기술한다.

2. GUI 테스팅 기술

GUI 테스트 도구들은 80년대 말에서 90년대 초 사이에 연구하기 시작했다.[3] 현재 GUI 자동 테스트에 사용되는 기술은 세 가지가 있다. 이 장에서는 Record-Palyback 기술, 명세 기반 테스트 기술 그리고 베타 테스트 기술에 대해 설명한다.

2.1 Record-Playback 테스팅

Record-Playback 기술은 사용자가 발생시키는 마우스 이벤트와 키보드 이벤트를 기록하고 이 기록한 이벤트를 재생하여 소프트웨어의 GUI 테스트에 사용한다.

현재 대부분의 GUI 테스트 도구들은 Record-Playback 기술을 적용 하며 Mercury Interactive 사에서 개발한 WinRunner가 그 전형적인 예로 들 수 있다.[3]

Record-Playback 기술은 단순한 패턴을 가지며 이 기술을 이용한 테스트 도구를 구현하기가 용의하다. 하지만 Record-Playback 기술은 3가지 중요한 문제점을 가지고 있다.

첫 번째, Record-Playback 과정은 노동집약적이다. 효과적인 테스터 케이스 생성하기 위해 테스터의 능력에 전적으로 의존한다[3]. 다양한 시나리오에 대한 테스트를 위해 많은 기록 과정이 필요하며, GUI의 변경 시 같은 작업을 반복해야 한다. 또한 이벤트 기록 시에 사소한 실수도 테스트 수행 시에 치명적이다.

두 번째, Record-Playback 기술은 단순히 마우스 이벤트와 키보드 이벤트의 내용만을 기록할 뿐, 기록된 테스트 케이스가 '어떤 일을 하는 지?', '어떻게 작동이 되는가?'에 대한 명세를 테스터가 따로 기술해야 한다. 만약 이러한 명세가 없다면 많은 양의 테스트 케이스를 감당하기 어려울 것이다.

세 번째, Record-Playback 기술에서 테스트 스크립트를 수정하는 것에 많은 노력이 필요하다[3]. 만약 시스템의 행동 명세가 변경된다면 처음 시스템의 실행 동안 기록된 테스트 스크립트가 더 이상 의미가 없을 수 있다. 시스템의 변화를 맞추기 위해 테스터는 시스템 명세에 맞추어 수동적으로 테스트 스크립트를 수정하거나 다시 이벤트를 기록해야 하는 번거로움이 있다.

2.2 명세 기반 테스팅

명세 기반 테스트는 시스템의 GUI 명세를 기반으로 시스템을 테스트한다. 명세 기반 기술 테스트는 다른 테스트 기술에 비교하여 볼 때 다음과 같은 이점을 갖는다[4].

- 테스트 명세는 테스트 케이스와 테스트 스크립트를 비교하기 때문에 예상되는 전체 시스템의 행동이나 기능을 간결한 방법으로 표현한다.
- 엄격한 명세 기반 테스트 수행은 요구사항이나 디자인 명세가 일괄적이고, 완벽하며, 분명하게 기술되는 것을 요구한다. 이것은 정적인 세부분석을 통해서 전형적이고 중요한 시나리오를 얻기 위한 기초를 제공한다.
- 시스템 명세는 다른 추상적인 면에서 예상되는 시스템 행동을 기술하는 것에 용이하다. 시스템의 정확한 행동이 명시적으로 기술되면 테스터는 시스템 명세에서 테스트 케이스를 조직적으로 얻을 수 있다.
- 테스트 범위 기준이 결정되어지면 테스트 케이스는 자동적으로 생성할 수 있다.

하지만 명세 기반 GUI 테스트는 높은 수준의 GUI 명세를 요구한다. 이런 명세를 얻는데 많은 노력이 필요하고 얻어진 명세를 목표 AUT(Application Under Test)에 연결하여 직접적인 테스트 수행이 어렵다[4].

2.3 베타 테스팅

베타 테스트는 GUI 테스트 중에 가장 인기 있는 방법이다. 베타 테스트는 베타 버전의 소프트웨어 복사본을 출시하여 일반 사용자들에게 테스트의 일부, 즉 GUI 테스트를 수행하게 하는 것이다. 그 예로 Microsoft 사는 Windows95 베타 버전의 소프트웨어 복사본을 약 400,000개를 출시하였다.[2]

GUI 테스트 방법 중 가장 많은 오류를 찾을 수 있는 방법이지만 비전문적인 테스터들로 구성되어 그 전문성이 떨어지고 많은 일반 사용자를 구하고 교육시키는 데에 어려움이 따른다.

3. Record-Playback기술 기반의 GUI 테스트 케이스 자동생성 방법

이 장에서는 Record-Playback기술을 이용하여 GUI 테스트 케이스 자동 생성에 대한 방법을 제시하고 그 예로 계산기의 테스트 케이스 생성을 기술한다.

3.1 테스트 케이스 자동 생성 방법

테스트 케이스 자동 생성 방법에 대해 논하기 전에 본 논문에서 사용되는 용어를 정의한다.

- 단위 사용자 이벤트(Unit User Event): 테스트 스텝을 구성하는 최소 단위의 작업 이벤트를 말한다. GUI의 각 요소에 최소 단위 이벤트를 기록함으로써 GUI의 변경에 대한 이벤트의 재기록을 최소화 할 수 있다. 예를 들어, A버튼을 클릭하고 B버튼을 클릭하고 C버튼을 클릭하는 테스트 스텝을 기록한 시나리오가 있다고 가정하자. 만약 어떠한 문제로 인해 "A->B->C"가 "B->C->A"로 작업 순서가 변경되었다면 시나리오는 더 이상 테스트에서 사용 할 수 없게 된다. 하지만 테스트 스텝을 구성하는 세 개의 단위 이벤트를 나누어 기록하면 이벤트의 순서가 바뀌어도 세 개의 단위 이벤트의 조합으로 테스트에 사용 할 수 있다.
- 정보 이벤트(Information Event): 단위 사용자 이벤트의 조합으로 생성할 수 있는 소프트웨어의 입력을 의미한다. 정보 이벤트는 테스트 수행 시 이벤트의 조합으로 다양한 입력을 생성한다.
- 명령 이벤트(Command Event): 하나의 단위 사용자 이벤트만으로 소프트웨어를 동작 시킬 수 있는 입력을 의미한다. 명령 이벤트는 테스트 수행 시 고정된 이벤트로 단일 입력을 생성한다.

일반적으로 소프트웨어는 외부로부터 명시된 일을 수행하기 위한 정보와 명시된 일을 수행시키는 명령에 대한 두 가지형태의 입력을 받는다. 첫 번째 입력은 정보 이벤트로 정의하고 두 번째 입력은 명령 이벤트로 정의한다. 정보 이벤트와 명령 이벤트를 나눔으로써 다양한 테스트 케이스의 자동 생성에 용의하다.

다음은 GUI 테스트 케이스 자동 생성을 위한 절차이다.

단계 1. 단위 사용자 이벤트 생성과 명세 작성

소프트웨어의 모든 GUI 구성에 대한 단위 사용자 이벤트를 생성하여 기록하고 이 이벤트에 대한 명세를 작성한다. 소프트웨어 GUI는 메뉴, 텍스트 라인, 버튼, Window, 네 가지 구성으로 분류한다. 네 가지 GUI 구성 요소는 서로 다른 사용자 이벤트 특성을 가지고 있다. 본 논문에서는 Window 대한 내용은 제외한다.

아래는 GUI 구성요소의 설명이다.

- 메 뉴 : 사용자가 최상위 메뉴를 클릭하면 메뉴는 하위 메뉴를 보여준다. 그리고 하위 메뉴를 클릭하면 다시 하위 메뉴를 가질 수도 있고 그렇지 않으면 클릭한 메뉴에 따라 소프트웨어에 명령을 수행시킨다. 최종 명령을 클릭하는 과정이 하나의 사용자 이벤트이다.
- 텍스트 라인 : GUI의 대부분은 마우스 이벤트에 의하여 활성화 되지만 텍스트 라인의 경우 키보드 이벤트에 의하여 활성화 된다.
- 버 튼 : 가장 단순한 사용자 이벤트를 가진다. 마우스 클릭에 의하여 활성화 된다.

테스터는 각 요소에 맞게 단위 사용자 이벤트를 생성하여 기록한다. 그리고 각 단위 사용자 이벤트 기록에 대한 명확한 명세를 작성한다. 명세가 명시되지 않으면 단위 사용자 이벤트 기록을 재사용하기 어렵다.

그림 1은 단위 사용자 이벤트 기록의 형태를 보여준다.

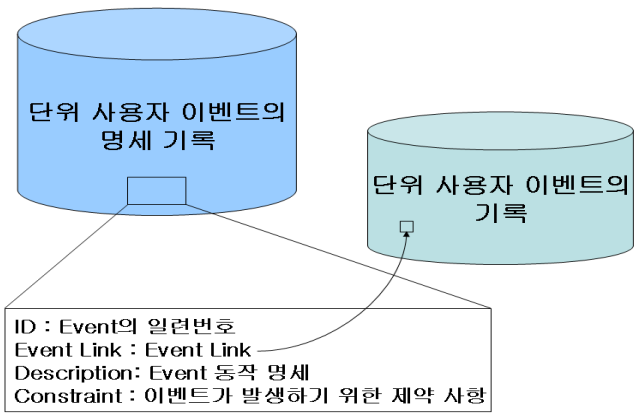

그림 1 단위 사용자 이벤트 저장 형태

단계 2. 단위 사용자 이벤트 그룹화

단위 사용자 이벤트를 정해진 조건에 따라 그룹을 형성한다. 각 특성과 목적에 따라 단위 사용자 이벤트의 그룹을 형성한다.

- 어떠한 GUI 구성 요소 인지?
- 키보드 이벤트를 이용한 것인가?
- 마우스 이벤트를 이용한 것인가?
- 데이터 입력에 대한 이벤트 인지?
- 소프트웨어의 동작에 대한 이벤트 인지?

그룹화는 자원의 재활용 증가 시키고 테스트 케이스 생성을 원활하게 할 수 있도록 돕는다.

단계 3. 시나리오 결정 후 정보·명령 이벤트 지정

테스트 시나리오를 결정하고 정보 이벤트와 명령 이벤트를 지정한다. 테스터는 시간이 적게 소요되고, 되도록 다양한 테스트를 수행할 수 있는 시나리오를 생성해야 한다.

아래는 시나리오 표현식을 정의한다.

[□:n, m] : □에 대해 n~m회 랜덤하게 재생
[■:n, m] : ■에 대해 n~m회 재생
[□|■] : □ 혹은 ■ 재생
[□]>[■] : 순차 재생
[ ]:테스트 스텝 □: 그룹 ■: 단위 이벤트

위 그룹에 속해 있는 단위 사용자 이벤트를 랜덤하게 선택한다.

시나리오가 결정되면 생성된 시나리오에 아래 사항을 고려하여 정보 이벤트와 명령 이벤트로 구성한다.

- 이벤트가 자주 변경되어야 하는지?
- 이벤트가 고정되어야 하는지?
- 어떠한 목적으로 사용되는 이벤트 인지?
- 유동적으로 변형되는 이벤트인지?
- 얼마나 자주 사용되는 이벤트 인지?

단계 4. 체크 포인트 모듈 삽입

테스트 진행 중에 체크 포인트 모듈이 위치할 곳을 선택한다. GUI 테스트 특성상 테스트 오라클과 최종 산출물이 일치하여도 테스트 수행 중 소프트웨어의 오류 발생이 없다고 확신할 수 없다. 그러므로 테스트 시나리오 내부에 오류 발생이 일어 날수 있는 위치를 선택하여 체크 포인트 모듈을 삽입한다. 삽입된 체크 포인트 모듈은 테스트 수행 시 화면을 캡처하여 테스트 로그 파일을 생성한다. 테스트가 수행이 끝난 후 테스터는 오류 발생 여부를 확인하기 위해 로그 파일을 분석한다.

단계 5. 테스트 스크립트 생성

단계 4를 바탕으로 테스트 케이스를 생성하고 테스트 스크립트를 생성한다.

3.2 계산기의 테스트 케이스 자동 생성

단계 1. 단위 사용자 이벤트 생성과 명세 작성

계산기의 모든 GUI 구성에 대한 소단위 사용자 이벤트를 생성하고 각 이벤트에 대한 명세를 작성한다.

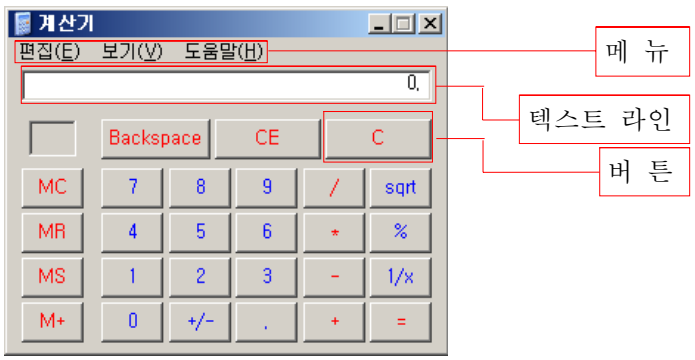

그림 2 계산기 GUI의 구성

그림 2와 같이 계산기의 GUI는 3가지 구성 요소를 가지고 있다.

처음으로 GUI 구성 요소 중 메뉴의 편집에 대한 단위 사용자 이벤트를 생성한다. 마우스로 편집을 클릭하고 하위 메뉴인 붙여넣기를 클릭하여 이벤트를 생성한다. 그리고 이 단위 사용자 이벤트에 대한 명세를 작성한다. 이런 식으로 각 메뉴에 대한 단위 사용자 이벤트를 생성한다. 그 다음으로 텍스트 라인에 대한 단위 사용자 이벤트를 생성한다. '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '+', '-', '*', '/', 'ENTER' 등을 생성하는 키보드 이벤트를 기록하고 각 이벤트에 대한 명세를 작성한다. 마지막으로 버튼에 대한 단위 사용자 이벤트를 생성한다. 각 버튼마다 마우스 이벤트를 생성하고 명세를 작성한다.

이 외에 더 많은 이벤트를 기록 되어야 하지만 본 논문에서 위 이벤트 예로 기술한다.

단계 2. 단위 사용자 이벤트 그룹화

단계 1에서 생성한 단위 사용자 이벤트 그룹을 생성한다. 각 GUI 구성요소에 따라 사용자 이벤트를 그룹 짓는다. 그리고 버튼과 텍스트 라인의 이벤트는 숫자를 생성하는 이벤트와 기호를 생성하는 이벤트로 구분한다. 그리고 마우스 이벤트와 키보드 이벤트로 나눈다. 계산기 이벤트 기록의 그룹은 그림 3과 같다.

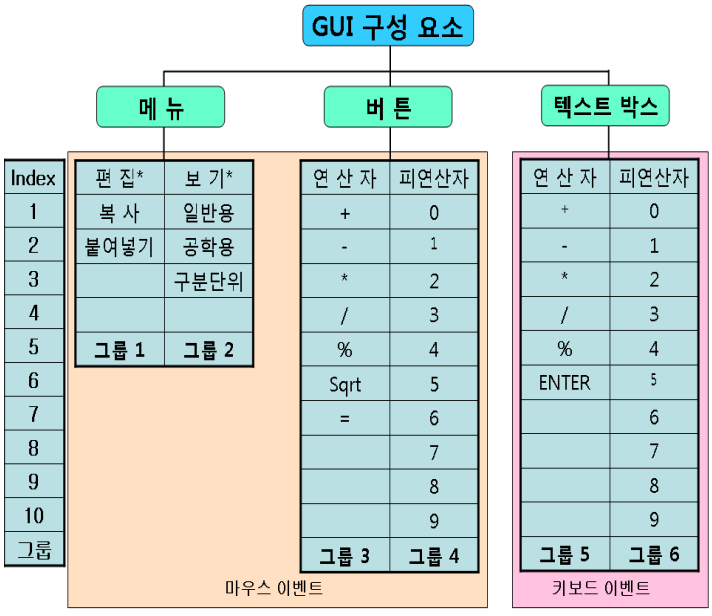

그림 3 단위 사용자 이벤트 그룹

단계 3. 시나리오 결정 후 정보·명령 이벤트 지정

계산기의 GUI 테스트 시나리오를 결정하고 정보 이벤트와 명령 이벤트를 지정한다.

• 테스트 시나리오를 결정

"0~99 + 0~9999 - 0~99 * 0~999 ="를 50번 반복하여 테스트 한다

• 시나리오 표현식

[[G6:1,2]>[G3(1):1,1]>[G6:1,4]>[G3(2):1,1]>[G4:1,2]>[G3(3):1,1]>[G 4:1,3]>[G5(6):1,1]:1,50]

다음은 시나리오 풀이로써 순서에 따라 시나리오가 수행된다.

① 그룹 6를 이벤트들을 재생하여 두 자리 숫자를 랜덤 하게 만든다.

② 그룹 3의 1번 이벤트를 선택하여 +를 이벤트를 재생한다.

③ 그룹 6의 이벤트들을 재생하여 네 자리 숫자를 랜덤 하게 만든다.

④ 그룹 3의 2번 이벤트를 선택하여 -를 이벤트를 재생한다.

⑤ 그룹 4의 이벤트들을 재생하여 두 자리 숫자를 랜덤 하게 만든다.

⑥ 그룹 3의 3번 이벤트를 선택하여 *를 이벤트를 재생한다.

⑦ 그룹 4의 이벤트들을 재생하여 세 자리 숫자를 랜덤 하게 만든다.

⑧ 그룹 5의 6번 이벤트를 선택하여 ENTER 이벤트를 생성한다.

⑨ 위 단계를 50번 반복 수행한다.

• 정보 이벤트와 명령 이벤트 결정

IE 1 = [G6:1,2]
CE 1 = [G3(1):1,1]
IE 2 = [G6:1,4]
CE 2 = [G3(2):1,1]
IE 3 = [G4:1,2]
CE 3 = [G3(3):1,1]
IE 4 = [G4:1,3]
CE 4 = [G5(6):1,1]

위 식에서 IE는 정보 이벤트이고 CE는 명령 이벤트이다.

단계 4. 체크 포인트 모듈 삽입

계산기의 GUI 테스트 진행 중에 체크 포인트 모듈이 위치할 곳을 선택한다. 시나리오가 5 +10 - 15 * 100일 경우 시나리오를 수행했을 때 최종 산출물은 0이다. 만약 '5 + 10 - 15' 까지 시나리오가 진행 되고 알 수 없는 오류에 의해 '* 100'이 수행 되지 않았더라도 최종 결과는 0이 된다. 그러므로 IE1 > CE1 > IE2 > CE2 > IE3 > '체크 포인트 모듈' > CE3 > IE4 > CE4 순서로 테스트를 수행해야 한다. 체크 포인트 모듈은 중간 테스트 수행 과정을 로그파일에 저장 한다.

단계 5. 테스트 스크립트 생성

단계 4를 바탕으로 계산기의 GUI 테스트 케이스를 생성하고 스크립트를 생성한다. 표 1은 단계 4의 시나리오 5 + 10 - 15 * 100을 XML로 작성한 테스트 스크립트의 예를 보여준다.

표 1 테스트 스크립트

3.3 테스트 케이스 자동 생성 방법의 장점

```
.....
<Sequence EventID = "G6(6)"/>
<Sequence EventID = "G3(1)"/>
<Sequence EventID = "G6(2)"/>
<Sequence EventID = "G6(1)"/>
<Sequence EventID = "G3(2)"/>
<Sequence EventID = "G4(2)"/>
<Sequence EventID = "G4(6)"/>
<Sequence EventID = "G3(3)"/>
<Check ModuleID = "M1"/>
<Sequence EventID = "G4(2)"/>
<Sequence EventID = "G4(1)"/>
<Sequence EventID = "G4(1)"/>
<Sequence EventID = "G5(6)"/>
.....
```

본 논문에서 GUI 테스트 케이스 자동 생성으로 Record-Playback 기술의 세 가지 문제점에 대한 개선이 이루어졌다. 첫 째로 이벤트 기록에 대한 노동집약적 문제의 개선이다. 테스트 스텝보다 더 작은 단위로 사용자 이벤트를 기록함으로써 중복되는 이벤트 기록을 최소화 하였다. 또한 이 기록들의 다양한 조합으로 양적으로 많은 사나리오를 생성 할 수 있고 최소 단위로 기록된 이벤트에 명세를 추가하여 테스트에 사용하는 이벤트에 대한 명확한 정보를 제공한다. 즉 이벤트에 링크된 명세는 이벤트와 이벤트들의 조합이 “어떤 일을 수행하는가?”를 설명한다. 이는 두 번째 문제인 이벤트 기록에 대한 부정확한 정보 제공을 개선하였다. 세 번째로 이벤트 그룹핑, 시나리오 표기법, 정보·명령 이벤트의 분류 등으로 인해 시스템의 명세 변경에 보다 유연하게 대처할 수 있다. 이는 변경된 시스템 명세에 한해서 이벤트를 새로 기록하거나 이벤트들의 조합을 새로 구성함으로써 다양한 테스트 스크립트 생성이 용이해졌다. 또한 테스트 수행 도중 발생 되는 오류 탐지의 필요성을 제시하였다.

4. 결 론

본 논문은 Record-Playback 기술을 이용하여 GUI 기반의 소프트웨어 테스트 케이스 자동으로 생성하는 방법을 제시 하였다. 테스트 스텝보다 더 작은 단위인 단위 사용자 이벤트로 이벤트를 기록함으로써 서론에서 언급한 GUI 테스트 수행의 어려움에 대한 전반적인 개선이 이루어졌다. 하지만 테스팅 검증에 대한 명확한 해결책을 제시하지 못하였다. 향후 본 논문에서 테스팅 검증에 대한 연구를 진행할 것이며, 제시한 GUI 테스트 케이스 자동 생성 방법을 구현하여 그 효과를 얼마만큼 얻을 수 있으며 구현 시 발생되는 문제점을 발견하여 분석하고 그 문제 대한 해결 방법을 연구 할 것이다.

참고문헌

[1] Atif M. Memon, Martha E. Pollack and Mary Lou Soffa, Hierarchical GUI Test Case Generation Using Automated Planning, IEEE Transactions on Software Engineering., vol. 27, no. 2, pp. 144-155, Feb. 2001

[2] Atif M.Memon, GUI Testing: Pitfalls and Process, IEEE Computer, pp.90-91, August 2002

[3] Automated GUI Testing, Tessella Support Services plc, January 1999, http://www.tssp.co.uk/Literature/Supplements/autogui.htm

[4] Jessica Chen and Suganthan Subramaniam, Specification-based Testing for GUI-based Applications, Software Quality Journal, 10, 205-224, 2002