시나리오의 자동 생성을 통한 GUI 테스트 케이스 생성 방법

(Test Cases Generation Method for GUI Testing with Automatic Scenario Generation)

이 정 규 † 국 승 학 †† 김 현 수 †††
(Jung Gyw Lee) (Seung Hak Kuk) (Hyeon Soo Kim)

요 약 최근 GUI 컴포넌트는 소프트웨어의 중요한 성공 요소이다. 따라서 GUI 컴포넌트는 반드시 검증되어야 한다. 그러나 소프트웨어의 GUI 검증을 위한 테스트에는 많은 시간과 비용이 소요된다. 이러한 자원의 소비를 줄이기 위해서는 GUI 테스트의 자동화가 필연적이다. 본 논문에서는 GUI 테스트를 수행하기 위해 고려해야 할 문제를 논하고 기존의 Record & Play-back 기술을 기반으로 한 GUI 테스트 케이스 생성 기법을 보완한다. 이를 위해 스파이 기술을 이용한 이벤트 생성 방법과 이렇게 생성된 이벤트를 그룹화하여 다양하고 효과적인 시나리오 생성 방법을 제안한다. 그리고 이렇게 생성된 시나리오를 바탕으로 GUI 테스트 케이스 생성 방법을 기술한다.

키워드 : GUI 테스팅, 테스트 케이스 자동 생성, 테스트 시나리오

Abstract In these days GUI components are recognized as the important driving elements to the successful software development. Thus they must be verified. In practice, however, GUI testing for verifying the GUI components needs lots of time and high costs. Test automation for GUI testing is a promising solution to save those expenses. In this paper, we discuss some issues considered to perform GUI testing and suggest a new method that improve the GUI test case generation method based on our previous 'record & playback' approach. For this, we suggest the event generation method using the 'spy' technique and the scenario generation method that generates effectively a lot of scenarios with the generated events. In turn we describe how to create GUI test cases from the generated scenarios.

Key words : GUI testing, automatic test case generation, test scenario

1. 서 론

최근 GUI(Graphical user interfaces) 컴포넌트는 사용자가 소프트웨어를 쉽게 작동할 수 있게 하며 사용자와 소프트웨어의 상호작용에 중요한 수단이 된다. 이러한 GUI 컴포넌트는 반드시 테스트가 되어 시스템의 안전성, 견고성 그리고 사용성이 검증 되어야 한다[1]. 하지만 GUI 컴포넌트 테스트는 GUI 컴포넌트의 단일 작동 테스트와 GUI 컴포넌트의 작동에 의한 소프트웨어의 내부 동작의 테스트가 병행되기 때문에 기존의 소프트웨어 테스트 과정보다 복잡하고 많은 양의 테스트 작업을 요구한다[2].

GUI 컴포넌트 테스트는 몇 가지 어려움이 따른다[2].

- 기존 소프트웨어 테스트는 코드의 양 또는 형태에 비례하여 테스트 노력이 요구된다. 하지만 GUI 컴포넌트 테스트는 테스트 코드와 무관하게 GUI 컴포넌트의

· 이 논문은 2008년도 충남대학교 학술연구비의 지원에 의하여 연구되었음
· 이 논문은 2008 한국컴퓨터종합학술대회에서 '시나리오의 자동 생성을 통한 GUI 테스트 케이스 생성'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 충남대학교 전기정보통신공학부 컴퓨터전공
jjangqwer@nate.com
†† 정 회 원 : 충남대학교 전기정보통신공학부 컴퓨터전공
triple888@cnu.ac.kr
††† 종신회원 : 충남대학교 전기정보통신공학부 컴퓨터전공 교수
hskim401@cnu.ac.kr

논문접수 : 2008년 9월 3일
심사완료 : 2008년 11월 14일

정보과학회논문지: 소프트웨어 및 응용 제36권 제1호(2009.1)

조합을 생성하며 양적으로 많은 테스트가 수행된다.

- 테스트 케이스 실행 단계에서 각 이벤트 단계가 명확하게 수행 되었는지 검증되어야 한다. 테스트 수행 중 이벤트의 오류가 테스트 케이스 전체의 오류로 검출되며 이러한 이벤트 오류의 검출은 쉽지 않다.
- GUI 컴포넌트는 프로토타이핑 모델로 개발된다. 이러한 개발과정과 소프트웨어의 잦은 버전 변화는 GUI 컴포넌트의 위치나 속성에 대한 변경을 유도하기 때문에 GUI 컴포넌트의 회귀 테스트는 자주 수행되어야 한다.

위와 같은 GUI 컴포넌트 테스트의 어려움은 테스터에게 GUI 컴포넌트 테스트를 수행하는데 많은 노력을 요구한다. 즉 수동적인 테스트 케이스의 생성, 유지, 평가는 많은 시간과 노력이 소비된다[2]. 특히 GUI 컴포넌트 테스트의 수행에 대한 양적인 면은 태스터에게 상당한 부담을 준다. 이러한 이유로 GUI 컴포넌트 테스트의 자동화는 필수적이다.

논문[3]에서는 Record-Playback 기반 기술에 바탕을 둔 테스트 케이스 생성 방법을 제시하여 GUI 컴포넌트 테스트에서의 문제점을 해결하고자 하였다. 그러나 테스트에 필요한 이벤트 생성 시 테스터의 개입으로 인하여 테스터에게 부담을 주는 단점이 있다. 본 논문에서는 테스터의 개입을 최소화하는 이벤트 생성 방법을 제시하고 이벤트 그룹화를 통해 단일 테스트 시나리오에서 다양하고 효과적인 테스트 시나리오를 자동으로 생성하는 방법과 이렇게 생성된 테스트 시나리오를 기반으로 GUI 테스트 케이스를 생성하는 방법을 제안한다.

이 논문의 2장에서는 GUI 컴포넌트를 제어하는 이벤트 생성 방법을 기술하고 3장에서는 이벤트 그룹화 방법과 시나리오 생성 방법을 기술한 후 4장에서는 3장에서 제안한 방법을 적용한 사례에 관하여 기술한다. 5장은 본 논문에서 제시한 방법을 수행하는 도구의 아키텍처에 대해 간단히 기술하고 6장은 결론 및 향후 연구를 기술한다.

2. 윈도우즈 기반 GUI 컴포넌트 이벤트 생성

이 장에서는 스파이 기술을 이용하여 윈도우즈 응용 프로그램의 GUI 컴포넌트 정보를 추출하는 방법을 기술한다. 이 논문에서는 윈도우즈 응용 프로그램에서의 GUI 컴포넌트를 컨트롤이라 부른다. 이 논문에서는 윈도우즈(Windows)와 윈도우(window)라는 용어를 함께 사용하고 있는데, 여기서 윈도우즈는 운영체제를 의미하며, 윈도우는 윈도우즈 응용 프로그램을 구성하는 컨트롤 중 하나이다.

2.1 윈도우 메시지

윈도우즈는 메시지 기반 운영체제로써 윈도우즈에서 일어나는 모든 행동을 윈도우즈에서 정의한 메시지로 제어하며, 메시지의 생성과 처리를 무수히 반복한다. 이

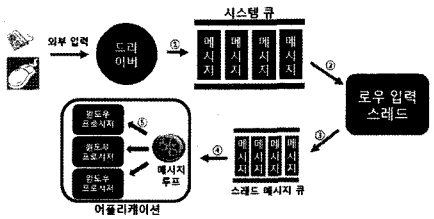

그림 1 윈도우 메시지의 처리 과정

러한 메시지를 윈도우즈 메시지라 부르며, 이 메시지는 응용 프로그램, 외부 입력, 시스템 내부 상황 변화 등을 제어하기 위한 여러 정보를 담고 있다.

한편, 윈도우를 조작하거나 제어하기 위해 사용하는 메시지를 윈도우 메시지라 부르며, 이 메시지는 윈도우즈 메시지에 포함된다. 그렇지만, 윈도우즈에서 발생하는 모든 메시지는 윈도우 메시지에 해당되지는 않는다. 왜냐하면 윈도우 메시지는 컨트롤들을 조작하거나 제어하기 위한 메시지로 한정되기 때문이다.

그림 1은 키보드와 마우스의 입력으로부터 생성된 윈도우 메시지의 생명주기를 보여준다[4]. 다음은 윈도우 메시지의 처리 순서이다.

① 외부에서 마우스 입력이 발생하면 마우스 드라이버는 메시지를 생성한다.
② 생성된 메시지는 시스템 큐에 삽입된다.
③ 로우입력 스레드는 시스템 큐의 메시지를 대상 어플리케이션의 스레드 메시지 큐에 삽입한다(스레드 큐는 각 응용 프로그램에 일대일 대응된다).
④ 어플리케이션의 메시지 루프는 스레드 메시지 큐의 메시지들을 해당 프로시저에 전달한다.
⑤ 윈도우 프로시저는 메시지를 처리한다.

응용 프로그램의 컨트롤 역시 윈도우 메시지에 의해 제어되며 테스터는 ①의 드라이버와 같이 컨트롤을 제어하는 메시지를 생성하여 테스트를 수행할 수 있다.

2.2 윈도우 메시지 생성

윈도우 메시지는 스파이와 후킹 기술로 생성할 수 있다. 이 절에서는 윈도우 메시지의 생성을 위하여 윈도우 메시지의 구성요소 대해 알아본다. GUI 컴포넌트를 구동시키기 위한 이벤트는 윈도우 메시지를 기반으로 생성된다.

2.2.1 윈도우 메시지

윈도우 메시지는 그림 2와 같은 메시지 구조체로 표현된다[4].

다음은 윈도우 메시지 멤버에 대한 설명이다.

㉮는 윈도우 핸들을 저장한다. 윈도우 핸들은 윈도우를 식별하기 위한 상수 값이다. ㉯는 메시지의 발송지를 식별하기 위해서 사용된다.

```
typedef struct tagMSG {
  ㉮HWND hwnd;
  ㉯UINT message;
  ㉰WPARAM wParam;
  ㉱LPARAM lParam;
  ㉲DWORD time;
  ㉳POINT pt;
} MSG, *PMSG;
```

그림 2 윈도우 메시지 구조체

㉯는 발생한 이벤트의 종류를 나타내는 '0'이상의 상수다. 윈도우즈에서는 이벤트의 종류를 나타내는 상수를 정의하고 있다. 예를 들면, 마우스의 왼쪽 버튼이 눌렸을 때는 WM_LBUTTONDOWN 이라는 상수가 저장된다.

㉰, ㉱는 메시지의 종류에 따른 전달인자를 저장한다. 예를 들어, 키보드 입력일 경우에는 키의 종류를 나타내는 값이 된다.

㉲, ㉳는 각각 메시지를 보냈을 때의 시각과 마우스 커서 좌표가 저장된다.

그림 1에서 드라이버는 윈도우 메시지의 멤버를 정의하고 윈도우 메시지를 생성한다.

2.2.2 스파이 기술 기반 윈도우 메시지 생성

윈도우 메시지를 정의 할 정보를 생성하거나 획득 할 수 있다면 우리는 그림 1의 드라이버처럼 윈도우 메시지를 생성하여 응용 프로그램의 컨트롤을 제어할 수 있다. Microsoft spy++와 같은 스파이 도구 혹은 스파이 라이브러리를 통해 컨트롤 제어에 필요한 정보를 획득할 수 있다.

윈도우즈 응용프로그램은 윈도우에 포장되며 이 윈도우를 이용하여 사용자로부터 입력을 받고, 작업을 수행한 결과를 출력한다. 윈도우는 하나의 컨트롤이다. 이러한 컨트롤은 윈도우즈가 운영체제 차원에서 제공하기 때문에 윈도우 클래스를 등록할 필요 없이 미리 등록되어 있는 윈도우 클래스를 사용하기만 하면 된다.

활성화된 윈도우는 핸들이라는 고유 번호와 고유 속성을 가진다. 그리고 모든 윈도우는 핸들로 관리되며, 바탕화면을 루트로 하는 트리 형태로 연결되어 있다.

윈도우의 핸들과 속성에 대한 정보는 윈도우즈가 제공하는 API를 통해 파악하는 방법과 후킹을 이용하여 파악하는 방법이 있다. 전자의 경우, 윈도우 핸들을 매개변수로 주면 윈도우의 속성 정보를 제공하는 API가 있으며, 또한 마우스의 위치 좌표, 윈도우 아이디(응용프로그램 안에서의 윈도우 고유 번호), 윈도우의 트리와 같은 속성 정보를 매개변수로 주면 윈도우의 핸들 정보를 제공하는 API가 있다.

후킹은 윈도우즈 시스템이나 응용 프로그램에서 발생되는 모든 메시지를 가로채는 것을 의미한다. 응용 프로

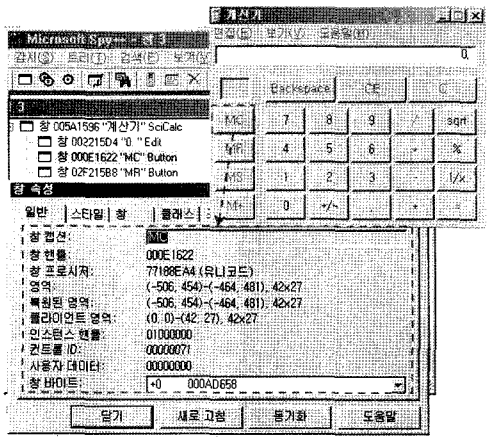

그림 3 컨트롤 속성 정보 추출 사례

그램의 실행과 동시에 윈도우도 생성된다. 이때 응용 프로그램은 윈도우즈에게 윈도우의 생성을 알리기 위해 윈도우 메시지를 발생시킨다. 이 메시지에는 윈도우의 핸들 정보가 담겨 있다. 후킹을 이용하여 윈도우의 핸들 정보를 획득하는 방법은 이 메시지를 가로채어 윈도우의 핸들 정보를 추출하는 것이다.

그림 3은 Microsoft spy++가 계산기의 'MC 버튼' 컨트롤의 속성 정보를 추출하는 사례이다.

이렇게 획득한 컨트롤 속성 정보를 통해 윈도우 메시지를 정의하여 특정 컨트롤을 제어하는 이벤트를 발생시킬 수 있다. 이벤트의 생성은 다음 장에서 기술한다.

3. 테스트 시나리오 생성 방법

이 장에서는 2장에서 소개한 윈도우 메시지를 이용하여 이벤트를 생성하고 논문[3]에서의 테스트 시나리오 생성 방법을 수정하고 보완한다.

논문[3]에서 단위 이벤트는 Record-Playback 기술로 생성되지만 이 논문에서 단위 이벤트는 스파이 기술로 생성된다.

- 단위 이벤트(Unit Event)

  단위 이벤트는 테스트 스텝을 구성하는 최소 단위의 작업 이벤트를 말한다. 단위 이벤트는 컨트롤을 제어하며 하나의 컨트롤에 대응된다. 이러한 단위 이벤트는 정보 이벤트와 명령 이벤트로 분류 된다. 예를 들어, "12 + 10 =" 시나리오의 경우, 6개의 단위 이벤트로 구성되고 '버튼 1', '버튼 2', '버튼 +', '버튼 1', '버튼 0', '버튼 =' 의 순서로 이벤트가 실행된다. '버튼 1'의 단위 이벤트는 두 번 실행된다.

  1) 정보 이벤트(Information Event)

  정보 이벤트는 응용 프로그램의 작동을 위해 필요한 데이타를 생성시키는 단위 이벤트를 의미한다.

테스트 수행 시 정보 이벤트들의 조합으로 다양한 입력 데이타를 생성한다. 예를 들어, 앞의 시나리오에서는 '12', '10'의 데이타 입력을 갖는다. 이러한 데이타 입력은 '버튼 1', '버튼 2', '버튼 0' 세 개의 단위 이벤트의 조합으로 생성된다.

2) 명령 이벤트(Command Event)

명령 이벤트는 응용 프로그램을 동작 시킬 수 있는 명령어에 해당한다. 예를 들어, 앞 시나리오에서는 '+', '='의 명령 입력을 갖는다. 이러한 명령 입력은 '버튼 +', '버튼 =' 을 제어하는 단위 이벤트에 의해 생성된다.

3.1 단위 이벤트 생성

논문[3]에서는 단위 이벤트를 생성하기 위하여 테스터가 발생시킨 이벤트를 기록한 다음 이 이벤트를 단위 이벤트로 생성한다. 이 방법은 Record-Playback 기술을 이용한다. Record-Playback 기술은 사용자가 발생시키는 마우스 이벤트와 키보드 이벤트를 기록하고 이 기록된 이벤트를 재생하여 응용 프로그램의 GUI 테스트에 사용한다. 현재 대부분의 GUI 테스트 도구들은 Record-Playback 기술을 적용한다[3]. 하지만 이 방법은 테스터의 개입을 증가 시키고 이벤트를 단순하게 기록하고 재생하기 때문에 하나의 GUI 테스트케이스만 생성된다.

이 절에서는 테스터의 개입을 최소화하고 다양한 테스트 케이스를 생성하는 단위 이벤트 생성 방법에 대해 기술한다.

그림 4의 정보 추출 모듈(Information Extraction module)은 컨트롤을 제어 할 수 있는 정보를 추출한다. 이 정보는 스파이 기술을 이용하여 얻은 컨트롤의 속성 정보로부터 생성된다. 하지만 그림 4의 ㉮버튼 정보에서 행동 상수 BN_CLICK은 버튼 컨트롤이 활성화 되었을 때 생성되므로 컨트롤 속성 정보에서 찾을 수 없다. 이러한 컨트롤의 행동 상수는 MSDN에 잘 정의 되어 있다. 우리는 컨트롤 속성 정보와 MSDN을 참조하여 컨트롤의 행동 상수를 얻을 수 있다. 각 컨트롤은 하나 이상의 행동 상수를 소유한다. 그리고 컨트롤의 행동 상수는 0개 이상의 인수(부가 정보)를 가질 수 있다. 예를 들면, 텍스트 박스에서 텍스트 입력의 경우 'WM_SETTEXT' 행동 상수는 스트링에 대한 포인터 정보가 필요하다. 포인터와 같은 정보는 미리 정해져 있지 않아 테스터의 개입이 요구된다. 단위 이벤트 설명 작성 시 테스터는 인수 정보처럼 자동으로 얻을 수 없는 정보를 작성한다.

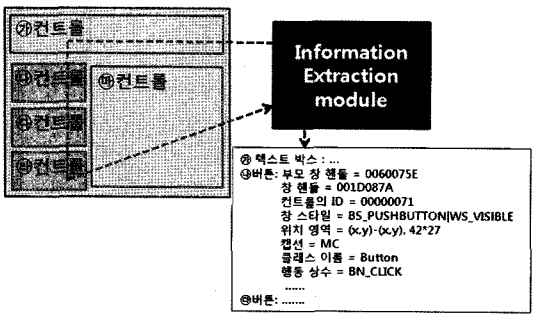

그림 4 스파이를 통한 컨트롤 정보 추출

그림 4의 정보 추출 모듈은 다음과 같은 작업을 수행한다.

- 스파이 기술로 응용 프로그램의 컨트롤 속성 정보를 추출한다.
- 추출 정보를 분석하여 윈도우 메시지 생성에 필요한 정보만을 필터링한다.
- 필터링한 정보를 분석하여 컨트롤 행동 상수를 나열한다.

테스터는 스파이 모듈에서 제공되는 정보를 참조하여 다음과 같은 설명을 작성한다.

- 컨트롤 행동 상수
- 행동 상수 인자
- 이벤트의 동작 설명
- 이벤트가 발생하기 위한 제약 사항
- 테스터를 위한 이벤트 설명

이벤트에 대한 동작 설명의 경우, 트리 컨트롤과 같이 복잡한 컨트롤은 컨트롤의 세부적인 동작에 맞추어 이벤트 설명을 기술한다. 이러한 설명은 이벤트를 효율적으로 관리하는데 도움이 된다. 그림 5는 단위 이벤트를 기술하기 위한 XML 템플릿이다.

이렇게 작성된 단위 이벤트는 SendMessage()와 같은 윈도우즈 API의 인자가 되어서 컨트롤을 제어하여 테스트를 수행한다. 함수의 인자는 윈도우 핸들, 통지코드, 컨트롤의 ID, 컨트롤의 윈도우 핸들 등이다. 그림 4에서 ㉯ 컨트롤의 정보로 SendMessage()를 채운다면 SendMessage(0060075E, WM_COMMAND, MAKELPARAM(BN_CLICK, 0000000071), 001D87A)가 된다.

```
<Event ID="이벤트 아이디">                                  이벤트 명세
  <Control>컨트롤 종류</Control>
  <Action Mar=Marco parm1=인수1>컨트롤 행동</Action>
  <Constraint>제약사항 </Constraint>
  <Description>이벤트 설명</Description>
  <property>                                               컨트롤 속성
    <parent>부모 창 핸들</parent>
    <handle>컨트롤 핸들</handle>
    <style>컨트롤 스타일</style>
    <location>위치 영역</location>
    <caption>컨트롤의 캡션</caption>
    <class>컨트롤 종류</class>
    <Macro>MARCO</Macro>
  </property>
</Event>
```

그림 5 단위 이벤트에 대한 XML 템플릿

3.2 단위 이벤트의 그룹화

3.1절에서 생성한 단위 이벤트를 아래와 같은 순서대로 그룹화 한다.

① 단위 이벤트를 키보드에 의해 제어되는 이벤트와 마우스로 제어되는 이벤트로 분류한다.

② ①에서 생성한 그룹의 단위 이벤트를 컨트롤의 종류에 따라 분류한다(컨트롤의 종류에는 메뉴 컨트롤, 버튼 컨트롤 등이 있다).

③ ②에서 생성한 그룹의 단위 이벤트가 데이타 입력에 해당하면 정보 이벤트로 분류한다.

④ ②에서 생성한 그룹의 단위 이벤트가 동작 명령 입력에 해당하면 명령 이벤트로 분류한다.

⑤ ④에서 명령 이벤트의 그룹에서 비슷한 명령을 가진 단위 이벤트들을 하위 그룹으로 분류한다.

①,②의 분류는 단위 이벤트 명세로 비교적 간단하게 분류할 수 있다. 하지만 ③,④,⑤의 분류는 정보 이벤트와 명령 이벤트의 분류이다. 정보 이벤트는 명령 이벤트에 비해 비교적 세밀하게 분류하지 않는다. 하지만 명령 이벤트는 응용 프로그램의 작동하기 위한 이벤트이기 때문에 보다 세밀한 분류가 필요하다. 이러한 이유로 ③,④,⑤의 분류는 단위 이벤트 명세만으로는 분류 할 수 없어 테스터의 개입이 요구된다. 테스터는 응용 프로그램의 명세를 참조하여 단위 이벤트를 분류한다. 위와 같은 단위 이벤트의 그룹화는 단위 이벤트를 효율적으로 관리할 수 있고, 다양하고 의미 있는 시나리오 생성을 할 수 있게 한다.

3.3 테스트 시나리오 생성

GUI 컴포넌트 테스트에서는 다양하고 많은 테스트가 수행되어 GUI 컴포넌트의 안전성, 견고성, 사용성을 검증하여야 한다. 일반적으로 테스터는 테스트 시나리오를 결정하고 그 시나리오를 바탕으로 테스트 데이타를 변경하여 GUI 테스트를 수행한다. 이와 같은 방법은 하나의 시나리오에서 데이타 값만을 변경하는 효과를 갖게 된다. 그러나 우리는 테스트 데이타뿐만 아니라 명령 컨트롤을 교체함으로써 보다 다양한 시나리오를 생성하고자 한다.

3.3.1 시나리오 정의

시나리오는 주어진 조건에서 대상 응용 프로그램의 외부 동작을 기술한 것이다. 즉 완성된 응용 프로그램이 외형적으로 나타난 기능을 수행할 때에 입력 조건과 그에 따른 반응을 기술한 것이다. 시나리오는 완성된 응용 프로그램의 기능 테스트에도 사용될 수 있다. 시나리오는 그 자체로 테스트 케이스로 사용된다[5].

3.3.2 시나리오 생성

시나리오는 다음과 같은 방법으로 생성된다.

① 기본 시나리오 생성

기본 시나리오 생성 방법은 다음과 같이 두 가지 방법이 있다.

1) 시나리오 작성: 테스터는 응용 프로그램의 명세를 참조하여 기본 시나리오를 명시적으로 작성한다. 예를 들어, 계산기의 경우 "3 + 5 ="와 같은 기본 시나리오를 작성한다.

2) 응용 프로그램 수행: 테스터는 응용 프로그램의 명세를 참조하여 응용 프로그램을 수행한다. 테스터가 수행한 시퀀스를 기록하여 기본 시나리오로 이용한다. 예를 들어, 계산기의 경우 '버튼 3', '버튼 +', '버튼 5', '버튼 =' 순서로 수행 했을 경우 이 수행 순서로 "3 + 5 ="와 같은 기본 시나리오가 기록된다.

② 정보 이벤트의 변화를 통한 시나리오 생성

기본 시나리오의 정보 이벤트를 대체하여 새로운 시나리오를 생성한다. 예를 들면, ①의 기본 시나리오에서 '3', '5'는 정보 이벤트이다. 테스트 도구는 '3', '5'가 속한 정보 이벤트 그룹에서 다른 정보 이벤트로 대체하여 새로운 시나리오를 생성한다. '3', '5'의 대체 이벤트가 '7', '6'이라면 새로운 시나리오 "7 + 6 ="이 생성된다.

③ 명령 이벤트의 변화를 통한 시나리오 생성

기본 시나리오의 명령 이벤트를 대체하여 새로운 시나리오를 생성한다. 예를 들면, ①의 기본 시나리오에서 '+'는 명령 이벤트이다. 테스트 도구는 '+'가 속한 명령 이벤트 그룹에서 다른 명령 이벤트를 대체하여 새로운 시나리오를 생성한다. '+'의 대체 이벤트가 '-'이면 새로운 시나리오 "3 - 5 ="가 생성된다.

위와 같이 정보 이벤트와 명령 이벤트의 변경으로 양적으로 많고 다양한 시나리오를 생성할 수 있다. 이렇게 생성된 시나리오는 단순한 이벤트들의 조합이 아닌 단위 이벤트들의 그룹화에 의해 의미 있는 테스트 시나리오가 된다.

3.3.3 시나리오의 표현

기본 시나리오는 시나리오 표현식으로 기술된다. 이때에 다음과 같은 시나리오 표현식을 사용한다.

다음 표 1은 여러 시나리오 표현식의 예이다.

- 표현식 [{G1.3}^{3}]은 그룹 1내에서 서브그룹 3에 속한 단위 이벤트를 임의로 선택하여 이벤트를 3회 생성한다. 이벤트를 임의로 선택하는 이유는 그룹 내의 특정 이벤트를 지시하는 인덱스를 표현하지 않았기 때문이다.
- 표현식 [G1(2)]는 그룹 1에 속한 단위 이벤트 2를 선택하여 이벤트를 하나만 생성한다.
- 표현식 [G4(2)|G2(1)]은 그룹 4에 속한 단위 이벤트 2나 그룹 2에 속한 단위 이벤트 1을 임의로 선택하여 이벤트를 생성한다.

표 1 시나리오 표현식

표현식	의 미
[ ]	테스트 스텝을 의미
( )n	()반복을 의미, n 반복 회수를 의미
Gx.y	이벤트의 그룹을 의미, x는 그룹의 아이디를 의미, y는 그룹의 서브 그룹아이디를 의미 예) G1, G2, G1.1, G2.1
Gx.y(Index)	단위 이벤트, 그룹의 특정 이벤트 의미 예) G1(1), G2(1), G1.1(3)
A | B	선택을 의미, A 이벤트 혹은 B 이벤트를 선택
>	다음 이벤트 생성

- 표현식 [G10(4)]>[{G2}^{2}]는 그룹 10에 속한 단위 이벤트 4를 한 번 생성하고, 그룹 2에 속한 단위 이벤트를 임의로 선택하여 2회 생성한다.

사실, 3.3.2 절의 '① 기본 시나리오 생성 단계'가 끝나면 생성된 시나리오는 파싱을 통해 시나리오 표현식으로 자동으로 변환된다. 예를 들어, "23 + 5 ="와 같은 기본 시나리오는 파싱을 통해 "[{G1}^{2}] > [G2.1] > [G1] > [G2]"와 같이 시나리오 표현식으로 변환된다. 이렇게 변환된 시나리오 표현식의 이벤트를 각 테스트 스텝의 그룹 내의 다른 이벤트로 교체하면 새로운 시나리오가 쉽게 생성될 수 있다. 즉, 위의 시나리오로부터 "16 + 8 ="과 같은 새로운 시나리오가 쉽게 생성될 수 있다. 여기서 이벤트들의 그룹은 고정되어 있지 않고 응용 프로그램에 따라 달리 정의된다. 구체적인 사례는 다음 장에서 자세히 기술한다.

3.4 체크포인트 삽입

테스트 진행 중에 문제가 발생할 수 있는 위치에 체크포인트를 삽입한다.

GUI 컴포넌트 테스트는 테스트 진행 중에 오류를 검출하기가 어렵고 테스트 자동화 도구의 한계로 테스터의 개입이 요구될 수 있다. 따라서 이 논문에서는 3.3 절에서 생성한 시나리오에 체크포인트를 삽입하여 테스트 중간에 발생할 수 있는 테스트 오류를 검출하고 테스터의 개입이 요청될 때를 대비한다.

체크포인트는 Ranorex[6] 테스팅 도구와 같이 텍스트 박스의 특징을 이용하여 계산기의 텍스트 박스 컨트롤을 테스트할 수 있다. 테스트 도구는 테스트 오라클과 텍스트 박스에 출력된 데이타를 추출하여 비교 검사한다.

이 논문에서는 테스트 스텝 중간에 체크 포인트를 삽입하여 테스트 진행 상황을 모니터링하고, 중간 결과를 확인한다. 일반적으로 GUI 테스팅의 경우 다양한 컨트롤들의 조합으로 많은 테스트 케이스가 발생하고 이들에 대한 최종 테스트 결과를 확인하기 위해 모든 테스트 오라클을 준비한다는 것은 현실적으로 불가능하다는 것이 알려져 있다[7].

3.5 테스트 스크립트 작성

시나리오 표현식을 바탕으로 테스트 스크립트가 생성된다. 테스트 스크립트는 그림 6과 같은 XML 문서로 표현된다.

```
<TestScript>
  <Sequence ID=시나리오 번호 times=반복횟수>          테스트 사이클
    <E1 GID=그룹번호 Index=그룹의 이벤트 번호 times=반복회수>
    </E1>
    <E2 GID=그룹번호 Index=그룹의 이벤트 번호 times=반복회수>
    </E2>
    <CheckPoint/>                    체크포인트
  </Sequence>
</TestScript>
```

그림 6 테스트 스크립트

4. 사례연구: 계산기

4.1 단위 이벤트 생성

계산기의 모든 컨트롤에 대한 정보를 추출하고 테스터를 위한 설명을 추가한다.

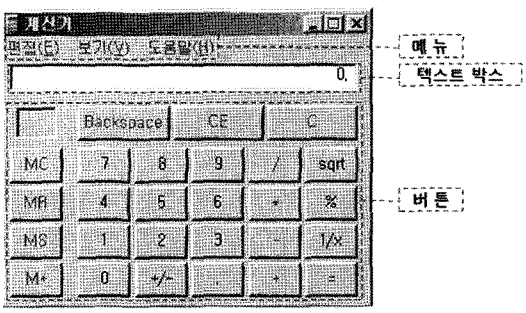

그림 7 계산기 GUI의 구성

그림 7과 같이 계산기는 세 종류의 컨트롤을 가지고 있다. 이 컨트롤에 대한 이벤트 생성은 컨트롤의 작동에 따라 다르게 생성된다.

- 버 튼 : 버튼 컨트롤에 대한 단위이벤트 생성의 경우, 테스터는 각 버튼이 무엇을 의미하는지에 대한 간단한 설명만 추가한다.
- 텍스트 박스 : 계산기의 텍스트 박스는 텍스트 박스가 활성화 되었을 때 사용자가 키보드를 입력할 경우, 데이타를 입력 받는다. 그리고 텍스트 박스는 계산기의 동작 결과 값을 출력한다. 테스터는 계산기의 명세를 참조하여 텍스트 박스에 허용되는 키보드 입력에 대한 단위 이벤트를 생성한다. 이것이 필요한 이유는 텍스트 박스 컨트롤의 경우 스파이 모듈에서 단지 텍스트 박스에 대한 속성 정보만을 추출함으로 키보드 입력에 대한 데이타 값의 정보를 얻을 수 없기 때문이다.
- 메 뉴 : 메뉴는 트리구조이다. 메뉴 트리에서 종단 노드를 제외한 나머지 노드는 메뉴의 종류를 구분하기 위한 것이다. 이는 종단 노드만이 응용 프로그램을

표 2 단위 이벤트 그룹

Grouping Table								
Control	Menu		Button			Text Box		
Group *	1	2	3		4	5		6
Index	편 집*	보 기*	연산자		피연산자	연산자		피연산자
1	복 사	일반용	Sub 1	Click '+'	Click '0'	Sub 1	Keydown '+'	Keydown '0'
2	붙여넣기	공학용		Click '-'	Click '1'		Keydown '-'	Keydown '1'
3		구분단위		Click '*'	Click '2'		Keydown '*'	Keydown '2'
4				Click '/'	Click '3'		Keydown '/'	Keydown '3'
5			Click '%'		Click '4'	Keydown'ENTER'		Keydown '4'
6			Click 'sqrt'		Click '5'			Keydown '5'
7			Click '='		Click '6'			Keydown '6'
8					Click '7'			Keydown '7'
9					Click '8'			Keydown '8'
10					Click '9'			Keydown '9'
이벤트	MOUSE					Keyboard		

제어할 수 있는 이벤트가 될 수 있음을 의미한다. 따라서 메뉴에 대한 단위 이벤트 생성은 종단 메뉴를 대상으로 생성한다.

계산기 컨트롤의 속성을 추출한 정보와 테스터가 작성한 설명이 XML로 기록된다. 그림 8은 XML로 기록된 단위 이벤트의 사례이다.

```
<Event ID="SciCalc_Button_0">
  <Control>Button</Control>
  <Action Mar=BN_CLICK parm1="">click button_MC
  </Action>
  <Constraint/>
  <Description>계산기의 버튼 컨트롤이며 메모리를 비운다.
  </Description>
  <property>
    <parent>0060075E</parent>
    <handle>001D087A</handle>
    <style>BS_PUSHBUTTON</style>
    <location>(x,y)-(x,y),42*27</location>
    <caption>MC</caption>
    <class>Button</class>
    <Macro>BC_CLICK</Macro>
  </property>
</Event>
```

그림 8 계산기의 단위 이벤트 사례

4.2 단위 이벤트 그룹화

4.1절에서 생성한 단위 이벤트의 설명에 따라 마우스와 키보드 이벤트를 분류한 후 컨트롤의 종류에 따라 단위 이벤트를 분류한다. 테스터는 계산기의 명세를 참조하여 아래와 같은 순서대로 각 그룹의 단위 이벤트를 정보 이벤트와 명령 이벤트로 분류한다.

① 정보 이벤트를 분류한다.

계산기에서 피연산자는 정보 이벤트로 분류 될 수 있다. 각 그룹에서 피연산자에 속해 있는 이벤트를 정보 이벤트로 분류한다.

② 명령 이벤트를 분류한다.

계산기에서 연산자는 명령 이벤트로 분류 될 수 있다. 계산기의 연산자 '+', '-', '*', '/', '%', 'sqrt', '='은 명령 이벤트로 분류된다. 이 명령 이벤트 그룹에서 사칙 연산자 '+', '-', '*', '/'는 하나의 서브 그룹으로 분류된다. 계산기의 메뉴에 속한 단위 이벤트도 편집과 보기의 그룹으로 분류한다.

표 2는 계산기에 대한 단위 이벤트의 그룹 테이블이다.

4.3 테스트 시나리오 생성

① 테스터는 계산기의 명세를 참조하여 기본 시나리오를 작성한다. 예를 들어, "12 + 1000 - 12 * 100 =" 이라는 시나리오를 기록한다.

② 기본 시나리오로부터 "[{G6}^{2}]>[G3.1]>[{G6}^{4}]>[G3.1]>[{G4}^{2}]>[G3.1]>[{G4}^{3}]>[G5]."와 같은 시나리오 표현식이 생성된다. 이 시나리오는 정보/명령 이벤트의 변화로 "0~99 +|-|*|/ 0~9999 +|-|*|/ 0~99 +|-|*|/ 0~999 +|-|*|/|%|sqrt="와 같은 시나리오가 생성될 수 있다.

③ 단계 ②에서 작성된 시나리오를 보다 의미 있는 시나리오로 생성하기 위해 테스터는 시나리오를 편집할 수 있다.

계산기에 대한 시나리오는 마지막 단계에 '=' 가 발생되어야 한다. 그러므로 위 시나리오의 [G5]는 "="에 대한 이벤트만 발생하는 표현식 [G5(6)]으로 수정한다. 그리고 테스터는 테스트 수행을 10번 반복하는 테스트 수행 회수를 지정한다. 이렇게 편집된 시나리오는 "{[{G6}^{2}]>[G3.1]>[{G6}^{4}]>[G3.1]>[{G4}^{2}]>[G3.1]>[{G4}^{3}]>[G5(6)]}^{10}"와 같다.

이 시나리오는 다음과 같은 의미를 갖는다.

㉮그룹 6에서 두 자리 숫자를 랜덤하게 발생시키는 이벤트 생성

㉮그룹 3의 서브 그룹 1에서 이벤트를 랜덤 생성
㉯그룹 6에서 세 자리 숫자를 랜덤하게 발생시키는 이벤트 생성
㉰그룹 3의 서브 그룹 1에서 이벤트를 랜덤 생성
㉱그룹 4에서 세 자리 숫자를 랜덤하게 발생시키는 이벤트 생성
㉲그룹 3의 서브 그룹 1에서 이벤트를 랜덤 생성
㉳그룹 4에서 세 자리 숫자를 랜덤하게 발생시키는 이벤트 생성
㉴그룹 5의 6번 이벤트인 'ENTER' 생성
㉵전체 시퀀스를 10번 반복 수행

4.4 체크포인트 삽입

계산기의 GUI 테스트 진행 중에 오류가 발생 할 수 있는 위치를 선택한다. 4.3 절의 시나리오에 의해 생성된 "5 + 10 - 15 * 100" 테스트 케이스의 결과는 0이다. 이 결과는 테스트 오라클 '0'과 비교하여 테스트가 제대로 수행 되었다고 볼 수 있다. 하지만 우리는 이 테스트가 올바르게 수행 되었다고 보증할 수 없다. 그 이유는 다음과 같다. 위 테스트 케이스가 '5 + 10 - 15' 까지만 수행되고 알 수 없는 오류에 의해 '* 100'이 수행 되지 않고 테스트 수행이 완료 되었다면, 이 테스트는 오류로 인한 테스트 실패가 보고되어야 한다. 하지만 테스트 중간에 이러한 오류를 확인 할 수 없는 상황에서 이 테스트의 결과는 '0'이 되어 테스트가 성공되었다고 보고된다. 그러므로 테스터는 '* 100" 전 단계에 체크포인트를 삽입하여 위와 같은 상황에 대비해야 한다. 체크포인트는 3.4 절에서 설명한 텍스트 비교 모듈이 될 수 있고 테스터의 직접적인 확인으로 이루어질 수 있다.

우리는 테스트 스크립트중간에 다음과 같은 "E1 > E2 > E3 > E4 > E5 > CheckPoint > E6 > E7 > E8" 체크포인트를 삽입할 수 있다.

4.5 테스트 스크립트 작성

4.4절을 바탕으로 계산기의 GUI 테스트 시나리오를 바탕으로 스크립트를 생성한다. 그림 9는 4.4절의 시나리오를 XML로 작성한 테스트 스크립트의 사례를 보여준다.

```
<TestScript>
  <Sequence ID=080415 times=10>
    <E1 GID=6 Index= times=2/>                    테스트 사이클
    <E2 GID=3.1 Index= times=1/>
    <E3 GID=6 Index= times=4/>
    <E4 GID=3.1 Index= times=1/>
    <E5 GID=4 Index= times=2/>
    <CheckPoint>                                  체크포인트
      <Textbox Option="RESULT">
        <TestOracle type="int" wait=5>0</TestOrcle>
      </Textbox>
    </CheckPoint>
    <E6 GID=3.1 Index= times=1/>
    <E7 GID=4 Index= times=3/>
    <E8 GID=5 Index=6 times=1/>
  </Sequence>
</TestScript>
```

그림 9 테스트 스크립트

5. 테스트 케이스 자동 생성 도구의 아키텍처

지금까지 테스트 케이스 생성 방법에 대해 알아보았다. 이 장에서는 테스트 케이스 생성 도구의 아키텍처에 대해 간단히 기술한다. 테스트 케이스 자동 생성 도구는 3가지 모듈로 구성되어 있다.

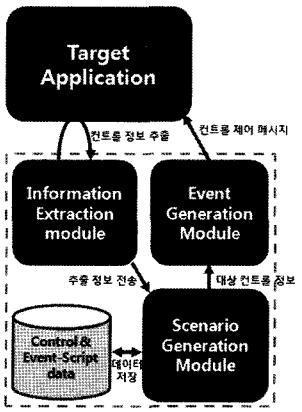

그림 10 테스트 케이스 자동 생성 아키텍처

1) Information Extraction 모듈
응용 프로그램의 GUI 컨트롤 정보를 추출하여 가공한다. 그리고 XML문서 형태로 Scenario Generation 모듈에게 정보를 제공한다.
2) Scenario Generation 모듈
테스터와 상호작용을 한다. 테스터가 추출정보(이벤트)의 설명 작성과 테스트 스크립트를 작성할 수 있도록 한다.
3) Event Generation 모듈
Event-Script Generation 모듈에서 작성 한 스크립트를 보고 대상 응용 프로그램의 GUI 컨트롤에 이벤트를 발생시켜 테스트를 수행한다.

6. 결론 및 토의

오늘날 GUI 컴포넌트는 소프트웨어에 중요한 역할을 한다. 따라서 GUI 컴포넌트는 반드시 검증 되어야 한다. 하지만 GUI 컴포넌트를 테스트하는 것은 테스터에게 많은 노력을 요구한다.

논문[3]에서 제안한 테스트 케이스 생성 방법은 단위 이벤트의 생성 과정에서 테스터에게 많은 개입을 요구하였다. 이 논문은 이러한 테스터의 개입을 줄여주는 단위 이벤트의 생성 방법을 제안하였다. 그리고 단위 이벤트를 그룹화하고 하나의 테스트 시나리오에서 같은 그룹 안에 단위 이벤트들을 교체함으로써 다양하고 효율적인 테스트 시나리오를 생성하는 방법을 제안하였다.

이 논문에서 제시하는 방법을 적용할 경우 소요되는 노력과 이러한 자동화 방법을 사용하지 않고 테스터가 직접 테스트 시나리오를 작성할 때의 노력을 정량적으로 측정해 보았더니 다음과 같은 결과를 관찰 할 수 있었다.

표 3 시나리오 생성 소요 시간 비교

시나리오 생성	시나리오 작성시간(초)	think time(초)	생성된 시나리오의 수	총소요 시간(초)
테스터	10	5	100	(10+5) ×100
자동화도구	10	0	100	10+0.01

위의 표에서 시나리오 작성 시간은 테스터의 경우 각각의 시나리오 작성마다 요구되는 시간이지만, 자동화 도구의 경우 처음 한 번만 시나리오를 작성하면 되므로, 첫 시나리오 작성에 소요되는 시간을 의미한다. 그리고 think time은 다음 시나리오를 작성하기 위해 잠시 생각하는 시간을 의미한다. 이 논문에서의 방법을 구현한 도구에서 시나리오 생성 시간을 측정하기 위해 도구 수행 시간을 프로파일링 한 결과 100개가량의 시나리오를 생성하는데 소요되는 시간은 대략 0.01초 정도 걸렸다. 물론 임의의 시나리오를 생성하게 하면 그 시간을 더욱 단축할 수 있지만 생성되는 시나리오가 중첩되지 않도록 수행하다보니 다소 시간이 소요되었다. 그렇지만 그 소요 시간은 테스터의 시간에 비교한다면 월등하게 줄어드는 것이 확실함을 알 수 있다.

이 논문에서 제안한 테스트 시나리오 생성 방법으로 다음과 같이 GUI 컴포넌트 테스트의 어려움을 개선한다.

첫 번째로 GUI 컴포넌트 테스트의 어려움 중 테스터에게 큰 부담이 되었던 시나리오 생성 과정에서 GUI 컴포넌트들을 임의로 조합하는 대신 분류된 이벤트를 선택적으로 조합함으로써 다양하고 의미 있는 테스트 시나리오를 대량으로 생성한다.

두 번째로 테스트 수행 중 발생되는 오류에 대한 문제는 테스트 케이스 내부에 오류가 발생할 수 있는 위치에 체크포인트를 삽입하여 감시한다.

세 번째로 소프트웨어의 변경으로 인한 GUI의 변경은 이 논문에서 제안한 방법에서는 큰 문제가 되지 않는다. 변경된 GUI 컴포넌트의 속성을 획득해 변경된 요소만을 변경 전의 GUI 이벤트에 적용하면 된다.

향후 우리는 이 논문에서 부족했던 체크포인트와 테스트 오라클에 대한 연구를 수행할 것이다.

참 고 문 헌

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

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

[3] 이정규, 김현수, 국승학, 조대완, "Record-Playback 기술 기반의 GUI 테스트 케이스 자동 생성", 한국정보과학회 2007 한국컴퓨터종합학술대회 논문집 제34권 제1호(B), pp. 96-100, 2007. 6.

[4] MSDN, http://msdn2.microsoft.com

[5] 김은주, 최은만, "시나리오를 이용한 객체 지향 시스템의 기능 테스트", 한국정보과학회 1996년도 가을 학술발표 논문집, 제23권 제2호(B), pp. 1523-1526, 1996. 10.

[6] Ranorex. www.ranorex.com

[7] Kai Y. Cai, Lei Zhao, Hai Hu and Chang H. Jiang, "On the Test Case Definition for GUI Testing," Proceedings of the Fifth International Conference on Quality Software(QSIC'05), pp. 19-28, 2005.

이 정 규

2007년 한남대학교 컴퓨터공학과 학사 2007년~현재 충남대학교 컴퓨터공학과 석사과정. 관심분야는 소프트웨어 테스팅, 소프트웨어 품질관리, 소프트웨어 아키텍처

국 승 학

2004년 충남대학교 컴퓨터과학과 학사 2006년 충남대학교 컴퓨터공학과 공학석사. 2006년~현재 충남대학교 컴퓨터공학과 박사재학중. 관심분야는 소프트웨어 테스팅, 소프트웨어 품질관리, SOA, 웹서비스

김 현 수

1988년 서울대학교 계산통계학과 학사 1991년 한국과학기술원 전산학과 공학석사. 1995년 한국과학기술원 전산학과 공학박사. 1995년~1995년 한국전자통신연구원 Post Doc. 1996년~2001년 금오공과대학교 조교수. 1999년~2000년 Colorado State University 방문교수. 2007년~2008년 Purdue University 방문연구교수. 2001년~현재 충남대학교 전기정보통신공학부 교수. 관심분야는 소프트웨어 테스팅, 소프트웨어 재공학, 컴포넌트 마이닝, 컴포넌트 테스팅, SOA