http://dx.doi.org/10.3745/KIPSTD.2011.18D.5.343

UIA 라이브러리를 이용한 GUI 테스트 자동화 방법

최 창 민[†] · 정 인 상[††] · 김 현 수[†††]

요 약

기존의 GUI 테스트 도구들은 테스트 케이스 준비 및 테스트 수행시 테스터의 개입을 많이 요구한다. 이러한 문제를 개선하기 위해 본 논문에서는 GUI 테스트 케이스를 구축할 수 있는 새로운 방법을 제안한다. 이 방법은 GUI 내에서 잠재적인 제어 흐름을 식별하여 GUI맵을 구성하는 방법이다. .NET Framework에서 제공하는 UIA 라이브러리는 GUI 컨트롤 정보를 추출하는 과정에 사용되며, 이를 통해 GUI맵을 구성한다. 추출된 GUI 컨트롤 정보를 기반으로 그룹화 매커니즘을 사용하여 테스트 시나리오를 생성한다. 다양한 테스트 시나리오는 어떤 GUI 컨트롤에 대해 그 컨트롤이 속하는 그룹에서 다른 컨트롤을 선택하여 그것을 대체함으로써 자동 생성된다. 기존의 GUI 테스트 도구들은 테스트 커버리지를 지원하지 않았지만, 이 논문에서 제안하는 방법은 GUI맵을 사용하여 실행되었거나 실행되지 않은 시나리오 흐름을 파악할 수 있으므로 이를 통해 테스트 커버리지를 측정할 수 있다.

키워드 : GUI 테스트, 테스트 케이스 자동생성, 테스트 시나리오, UIA 라이브러리

An automation method for GUI test using a UIA library

Choi Chang Min[†] · Chung In-Sang[††] · Kim Hyeon Soo[†††]

ABSTRACT

When preparing test cases and running the test the existing GUI test tools require many tester's interventions. To cope with such problem this paper suggests a new method to build test cases for GUI test. This method identifies the potential control flows within the GUI and constructs the GUI map. The UIA library in .NET Framework is used to extract information about the GUI controls and the GUI map is constructed by the extracted information. Test scenarios are generated from the extracted information about the GUI controls using the grouping mechanism. Based on the grouping mechanism, various test scenarios which are test cases in GUI tests can be made by replacing a GUI control by another one in the same group. The existing GUI test tools do not support the concept of test coverage. Since, however, our method survey which part of the GUI map is executed or not during running the test, the test coverage can be measured by using the GUI map.

Keywords : GUI Test, Test Case Automatic Generation, Test Scenario, UIA Library

1. 서 론

GUI(Graphical user interface) 기반의 응용 소프트웨어는 외부 이벤트를 처리할 수 있는 메뉴 및 다양한 컨트롤을 포함하고 있으며 사용자들은 이를 사용하여 소프트웨어를 구동시킨다. 이러한 GUI 메뉴 및 컨트롤들은 소프트웨어를 동작시키는 중요한 수단이기 때문에 반드시 테스트 되어야 하며 이를 통해 소프트웨어의 안전성, 견고성 그리고 사용성을 높일 수 있다[1]. 하지만 GUI 테스트는 단일 동작 테스트와 소프트웨어의 내부 동작 테스트가 병행되기 때문에 기존의 소프트웨어 테스트 과정보다 복잡하고 많은 양의 테스트 작업을 요구 한다[2].

GUI 프로그램은 이벤트 기반으로 동작한다. 따라서 동일한 UI구조라고 할지라도 사용자가 발생시키는 이벤트 순서에 따라 다양한 시나리오가 발생할 수 있으며, 이를 모두 테스트하기 위해서는 테스터의 많은 노력이 필요하다. 이러한 GUI 테스트의 어려움들을 정리하면 다음과 같다.

- GUI 프로그램에는 다양한 기능의 메뉴 및 컨트롤이 존재하기 때문에 이를 테스트하기 위해서는 많은 테스트가 수행되어야 한다.
- 이벤트 방식으로 구동하기 때문에 하나의 기능 실행 시 다수의 컨트롤들의 조합이 발생할 수 있다. 따라서 테스트를 수행하기 위해서는 해당 조합을 만족시킬 수 있는 많은 양의 시나리오를 생성해야 한다.

※ 이 연구는 2010년도 충남대학교 학술연구비에 의해 지원되었음.
† 준 회 원 : 충남대학교 컴퓨터공학과 박사과정
†† 정 회 원 : 한성대학교 컴퓨터공학과 교수
††† 종신회원 : 충남대학교 컴퓨터공학과 교수(교신저자)
논문접수 : 2011년 6월 8일
수 정 일 : 1차 2011년 7월 4일
심사완료 : 2011년 7월 7일

- GUI 프로그램은 다수의 윈도우로 구성되며 상호간의 관계를 가지고 있는 복잡한 구조이다. 하나의 윈도우에서 발생한 이벤트는 다른 윈도우의 특정 컨트롤에게 영향을 줄 수 있다. 즉, 윈도우 상호간 관계를 고려해서 테스트가 수행되어야 하기 때문에 프로그램의 정확한 호출 구조를 이해하여야 한다.

GUI 프로그램도 결국 하나의 소프트웨어 프로그램이기 때문에 다음과 같이 소프트웨어 테스팅이 가지고 있는 어려움들도 존재한다.

- 생성된 많은 테스트 시나리오를 실행하고 실행 결과를 분석하는 과정은 많은 시간과 노력이 요구된다.
- 프로그램에서 발생하는 오류 중 복합적인 연동에 의한 오류일 경우 다시 재현하기 어렵다.
- 프로그램은 개발과정이나 배포 이후 소프트웨어의 잦은 버전의 변화가 발생한다. 이러한 변화가 발생하게 되면 기존에 작성하였던 많은 양의 테스트 시나리오의 수정이 요구된다.

이러한 GUI 테스트의 어려움들을 해결하고자 많은 연구와 테스트 자동화 도구들이 개발되었다. 기존 GUI 테스트의 연구 및 자동화 도구들은 주로 Record-Playback 기반 기술과 명세 기반 기술로 나뉠 수 있다. Record-Playback 기반 기술은 테스터가 테스트 대상 프로그램을 동작시켰을 때 발생한 이벤트를 저장한 후 이것을 테스트 케이스로 사용하여 저장된 이벤트를 재생하는 방법으로 테스트를 수행한다. 저장된 이벤트를 재생할 때 입력정보를 수정함으로써 수동으로 테스트 시나리오를 확장하게 된다. 명세 기반 기술은 테스트 대상 응용 프로그램의 명확한 이해를 요구한다. 또한 명세 기반 기존 도구들은 기존의 프로그래밍/스크립트 언어 환경에서 테스트 스크립트를 작성하기 때문에 테스터에게 고급언어에 대한 전문적인 지식을 요구한다. Record-Playback 기반 기술과 명세 기반 기술은 나름대로의 장점을 갖고 있지만 두 방법 모두 테스트 케이스 생성 과정에서 테스터에게 많은 부담을 준다.

이 논문에서는 그룹화 기법을 이용한 테스트 자동화 기법을 제시한다. 이 기법은 GUI 컴포넌트를 제어하는 단위 이벤트를 추출하고 이것을 그룹화한다. 테스트 케이스 생성 과정에서 하나의 테스트 시나리오에서 같은 그룹 내의 단위 이벤트들을 교체함으로써 테스터가 원하는 만큼의 의미 있는 테스트 시나리오를 자동으로 생성하고, 생성된 시나리오를 가지고 자동으로 테스트를 수행한다. 또한 GUI MAP을 기반으로 테스트 수행 후 커버리지에 대한 판단도 자동화 할 수 있다. 특히, 테스트 케이스 생성 과정을 자동화하기 위하여 .NET에서 지원해주는 UIA(User Interface Automation) 라이브러리를 사용하였다.

논문의 구성은 다음과 같다. 2장에서는 GUI 테스트와 관련된 기존 연구들을 기술하고 3장에서는 논문에서 제시하는 테스트 자동화 방법을 기술한 후 4장에서는 3장에서 제안한 방법을 적용한 사례에 관하여 기술한다. 5장에서는 결론 및 향후 연구를 기술한다.

2. 관련 연구

2.1 GUI 테스트 기법

GUI 테스트 기법들은 1980년대 말에서 1990년대 초 사이에 연구하기 시작했으며, 현재 GUI 테스트 자동화에 사용되는 기법으로 Record-Playback 기술 기반 테스트 기법, 명세 기반 테스트 기법 등이 있다[3]. 이 절에서는 일반적으로 GUI 테스트 자동화에 사용되는 기법과 이 논문이 개선하고자 하는 그룹화 기법에 대해 살펴본다.

2.1.1 Record-Playback 기술 기반 테스트 기법

Record-Playback 기술은 사용자의 마우스 및 키보드 입력을 통해 선택된 GUI에서 발생한 이벤트를 기록하고 기록된 이벤트를 재생하여 소프트웨어의 GUI 테스트를 수행하는 방법이다.

현재 대부분의 GUI 테스트 도구들은 Record-Playback 기술을 적용하고 있으며, Record-Playback 기술은 단순한 패턴을 가지고 있어 이 기술을 이용한 테스트 도구를 구현하기가 용이하다. 하지만 Record-Playback 기술은 다음과 같은 중요한 문제점을 가지고 있다.

- Record-Playback 기술을 이용한 테스트 케이스 생성은 테스터의 행동을 저장하여 생성하기 때문에 테스터에게 부담을 준다. 다시 말해, 다양한 시나리오를 생성하기 위해 많은 기록 과정이 필요하며, GUI의 위치 변경 시 같은 작업을 반복해야 한다. 또한 이벤트를 기록하는데 사소한 실수도 테스트를 수행하는데 치명적일 수 있다.
- Record-Playback 기술로 저장된 테스트 케이스는 테스트 케이스에 대한 행동이 명시 되어야 한다. Record-Playback 기술 특성상 테스트 케이스의 이벤트가 발생되기 이전에는 대상 테스트 케이스가 어떻게 동작하는지 알 수 없다. 만약 이러한 명세가 없다면 많은 양의 테스트 케이스를 효율적으로 관리 할 수 없다.

2.1.2 명세 기반 테스트 기법

명세 기반 테스트는 시스템의 GUI 명세를 기반으로 소프트웨어를 테스트한다. 명세 기반 테스트는 다른 테스트 기술과 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 데에 용이하다. 시스템의 정확한 행동이

명시적으로 기술되면 테스터는 시스템 명세에서 테스트 케이스를 조직적으로 얻을 수 있다.

- 테스트 범위 기준이 결정 되어 지면 테스트 케이스는 자동으로 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.

2.1.3 그룹화 테스트 기법

그룹화 테스트 기법은 Record-Playback 기술의 장점과 명세기반 기법의 장점을 취하여 이벤트의 추출과 실행은 Record-Playback 접근 방법, 추출된 이벤트를 그룹화하는 과정은 명세 기반 기법을 적용함으로써 다양하고 효과적인 테스트 시나리오를 자동으로 생성하는 방법이다. 이 기법은 GUI 컴포넌트를 제어하는 단위 이벤트를 생성하고 단위 이벤트를 그룹화한다. 테스트 수행 시 하나의 테스트 시나리오에서 같은 그룹 내의 단위 이벤트들을 교체함으로써 다양하고 효율적인 시나리오를 생성한다[5].

그룹화 테스트 기법은 기존 테스트 기법의 수동적인 시나리오 확장과 달리 그룹으로 분류된 이벤트를 선택적으로 조합함으로써 다양하고 의미 있는 테스트 시나리오를 대량으로 생성한다는 장점을 가지고 있다. 하지만 다음과 같은 문제점도 가지고 있다.

- 프로그램 실행 중 동적으로 생성되는 윈도우에 대한 시나리오의 생성은 불가능하다. 이는 최초 프로그램이 실행될 때 획득한 정보만을 가지고 이벤트 그룹화를 수행하고 실행 중에 동적으로 생성되는 윈도우 및 컨트롤들에 대한 정보를 획득하지 못하기 때문이다.
- 그룹화 및 단위 이벤트 생성 시 테스터에게 복잡한 작업을 요구한다.
- 그룹화 기법은 단일 윈도우 환경만을 고려하였다. 따라서 멀티윈도우 환경에서의 테스트 처리가 불가능하다.

이러한 문제점들을 해결하고자 제시된 연구가 참고문헌 [9]에서 제시된 방법이다. 참고문헌 [9]의 방법은 기본적으로 참고문헌 [5]의 방법과 개념적으로 동일한 그룹화 테스트 기법으로써 위에 나열한 문제점들을 개선하는데 초점을 맞추었다. 그렇지만 참고문헌 [9]에서 제시한 방법도 다음과 같은 문제점을 여전히 갖고 있다.

- 프로그램 실행시 모든 컨트롤의 정보를 획득할 수 없다는 문제점을 가지고 있다. PE Format 분석 방식을 통해 프로그램이 가지고 있는 모든 UI정보를 획득할 수 있지만, 사용자가 만든 컨트롤, DLL로 구성되어진 컨트롤 등은 획득할 수 없다.
- 구현이 어렵다. PE Format 분석 과정이 필요하다.
- 그룹화 과정에서 테스터가 수행하는 작업량은 기존 방식에 비해 줄어들었지만, 아직도 많은 부분들을 요구한다.

2.2 기존 GUI 자동화 도구

앞에서 GUI 테스트 기법에 대해 살펴보았다. 이 절에서는 기존의 GUI 컴포넌트를 테스트 하는 자동화 도구에 대해 살펴본다. 기존의 GUI 테스트 자동화 도구는 앞에서 설명한 Record-Playback 기술 기법과 명세 기반 기법을 상호 보완해서 사용하고 있다.

〈표 1〉 GUI 테스트 도구의 특징 및 단점

	특징	단점
Test-Complete3[6]	- 테스트 스크립트를 이용해 작성 - 테스트 스크립트 작성을 위한 GUI 도구를 제공 - 스크립트의 수정을 통해 시나리오 확장 - 명세를 통한 시스템 행동 분석 - Recording & Playback 기능 지원 - 명세 기반에 Playback 기술 적용	- 스크립트에 대한 이해를 요구 - 스크립트 작성 이전에 프로그램의 명세를 정확이 이해해야 함 - 동일한 패턴의 시나리오를 가지고 Input 데이터를 변경할 경우 수작업을 필요로 함
Ranorex Recorder[7]	- Recording을 통해 시나리오를 자동생성 - GUI환경을 통해 사용자가 도구를 쉽게 사용할 수 있도록 구성됨 - 저장된 시나리오는 스크립트 형태로 변환되어 저장 - Recording & Playback 기능 지원 - Playback 기술에 명세기반 기술 적용	- Recording환경과 Play환경의 변화가 생기면 부정확한 Play 발생 - 수작업을 통한 스크립트 변경을 통해 시나리오를 확장
그룹화 테스트 방법[5]	- 이벤트 선택적 조합이 가능 - 하나의 테스트 시나리오를 기반으로 다양한 시나리오 자동 생성	- 그룹화를 위한 복잡한 작업을 요구 - 그룹화에 필요한 객체 정보의 동적 획득 불가 - 단일 윈도우 환경만 고려 - Recording 지원이 안 됨 - 자동 생성된 시나리오의 커버리지 문제
그룹화 테스트 개선 방법[9]	- 이벤트 선택적 조합이 가능 - 하나의 테스트 시나리오를 기반으로 다양한 시나리오 자동 생성 - 멀티 윈도우 환경 지원	- 완벽한 UI정보를 획득하기가 어려움 - 구현이 여려움 - 그룹화를 위한 복잡한 작업 요구 - 자동 생성된 시나리오의 커버리지 문제

<표 1>에 보면 TestComplete6[6]은 명세 기반 기법을 이용하여 테스트 케이스를 생성하며, 명세 기반 기법의 부족한 부분을 Record-Playback 기술 기법으로 보완한다. 명세 기반 기법을 이용하는 다른 도구와 마찬가지로 TestComplete6 역시 테스트 스크립트를 작성하기가 쉽지 않다. 이러한 이유로 TestComplate6은 테스트 스크립트 작성에 있어서 어려움을 덜어주기 위해 다양한 기능을 GUI로 제공한다. 하지만 이러한 GUI 들은 테스터가 테스트 스크립트를 직접 작성하는 것보다 많은 행동을 요구한다.

Ranorex[7]는 TestComplete6와 반대로 Record-Playback 기술을 이용하여 테스트 케이스를 생성하고 Record-Playback 기술의 부족한 부분을 명세 기반 기법으로 보완하여 테스트 케이스를 생성한다. Ranorex의 GUI는 사용자가 쉽게 도구를 사용할 수 있게 구성되어 있다.

그룹화 테스트 방법은 테스터의 개입을 최소화하는 테스트 케이스 생성 방법과 테스트 케이스를 효율적으로 유지하는 방법을 제시하기 위해 고안되었다[5]. 테스트 케이스는 시나리오의 자동 생성을 통해 생성되지만, 이를 위한 선행 작업으로 GUI 컴포넌트를 제어하는 이벤트 생성과 이벤트의 그룹화가 필요하다. 이 방법은 의미 있는 테스트 시나리오를 자동 생성하지만, 이를 위해 또 다른 복잡한 선행 작업을 요구한다.

3. GUI 테스트 자동화 방안

소프트웨어 테스트를 위해 테스트 자동화 도구를 사용하는 주된 이유는 매뉴얼 테스트가 가진 여러 가지 단점을 극복하기 위해서다. 앞선 연구에서 자동화 테스트와 비교하여 매뉴얼 테스트가 두 배 이상의 시간과 비용을 요구한다는 사실을 제시하고 있다[8]. 테스트 자동화 도구의 성능은 주어진 테스트 프로세스의 각 단계를 얼마만큼 정확하게 자동화하느냐에 달려 있다.

이 절에서는 2절에서 분석한 테스트 방법 중 그룹화 테스트 방법[5, 9]을 개선한 GUI 테스트 자동화 방법을 제안한다. 테스트 단계 중 테스트 설계, 수행, 결과 분석 부분에서 자동화를 수행할 수 있으며 이 논문에서는 아래의 부분들에 대한 자동화에 초점을 맞추어 기존 방법[9]을 개선한다.

- 테스트 케이스 생성
- 테스트 케이스 실행
- 시나리오 관점에서의 커버리지 판단

다음은 이 논문에서 제시하는 테스트 자동화 시나리오이다.

① GUI MAP을 구성한다. 이를 위해서 단위 객체의 정보를 획득하여야 하며 단위 객체의 특성 및 단위 객체 사이의 관계를 이해하여야 한다.
② 그룹화 및 입력 데이터를 설정한다.
③ 테스터가 입력한 기본 시나리오와 그룹화 된 결과물을 이용하여 의미 있는 다양한 시나리오를 생성한다.
④ 테스트를 수행한다. 이 때 수행된 결과 이미지를 객체 단위로 저장하게 되고, GUI 맵을 구성하고 있는 객체들의 커버리지를 체크한다.

3.1 관련 요소 정의

이 절에서는 논문에서 제시하는 방법에 사용되는 관련 요소들을 정의한다.

3.1.1 GUI MAP

GUI MAP은 테스트 대상 프로그램이 가지고 있는 GUI 단위 객체들을 트리 형태로 표현한 것이다. 트리구조의 구성 관점은 이벤트 발생 흐름에 있다. 예를 들어 'B 메뉴'를 클릭함으로서 'A 팝업 윈도우'가 생성된다면 'B메뉴'의 자식 노드에 'A팝업 윈도우'가 위치한다. 따라서 GUI MAP을 분석하면 어떠한 이벤트 흐름으로 해당 객체에 접근하게 되는지 쉽게 이해할 수 있다.

3.1.2 단위 객체(Unit Object)

단위 객체란 독립적으로 이벤트를 발생 시킬 수 있는 최소 단위 GUI 요소이다[10]. 프로그램의 구성요소 중 각각의 메뉴 아이템과 GUI 컨트롤들은 사용자와 상호작용하는 주요한 부분이고 이를 통해 프로그램에 이벤트가 전달되게 된다. 단위 객체의 예를 들면, 모든 프로그램이 공통적으로 가지고 있는 파일(F), 저장하기(S) 등의 메뉴 아이템과 "에디트", "버튼" 등의 컨트롤이 있다.

예를 들어 계산기 프로그램에서 "1 + 2 = " 시나리오의 경우, 4개의 단위 객체로 구성되고 '버튼 1', '버튼 +', '버튼 2', '버튼 = '의 순서로 단위 객체가 호출된다. 이러한 단위 객체는 정보 객체, 명령 객체, 뷰어 객체, 팝업 객체로 분류된다.

- 정보 객체(Information Object)

정보 객체는 응용 프로그램의 작동을 위해 필요한 데이터를 생성시키는 객체이다. 예를 들어 체크박스 컨트롤(대/소문자 구분)과 라디오 버튼(위로, 아래로) 등이 정보 객체에 해당한다. 이 컨트롤들은 프로그램 구동과 관련된 이벤트를 발생시키지 않고 보이지 않는 정보를 수정함으로써 프로그램 진행에 영향을 주게 된다.

- 명령 객체(Command Object)

명령 객체는 응용 프로그램을 구동 시키는 객체이다. 예를 들어 버튼 컨트롤이 대표적인 명령 객체에 해당된다. 이 객체는 이벤트를 발생시켜 프로그램을 동작시키게 된다.

- 뷰어 객체(Viewer Object)

뷰어 객체는 사용자에게 정보를 보여주는 객체이다. 일반적으로 정보 객체나 명령 객체는 마우스 이벤트로 동작하지만 뷰어 객체는 키보드 입력에 의해 동작한다. 예를 들어, 계산기의 입력 정보를 보여주는 에디트 컨트롤이 대표적인 뷰어 객체라 할 수 있다. 즉 뷰어 객체는 프로그램 구동 시 발생되는 정보를 보여주는 객체이다.

- 팝업 객체(PopUp Object)

팝업 객체는 자식 윈도우를 생성하는 객체이다. 예를 들

어 특정 메뉴는 다른 팝업 윈도우를 생성하게 된다. 이러한 객체를 팝업 객체라 할 수 있다.

3.2 제안하는 GUI 테스트 자동화 기법

이 절에서는 논문에서 제안하는 자동화 기법을 설명한다. 첫 번째로 단위 객체 추출 방식을 설명하고, 두 번째로 추출된 정보를 이용한 GUI MAP구성 방법을 설명한다. 세 번째로 단위객체의 특성에 따른 그룹화 및 기본 정보 설정 방법을 설명하고, 네 번째로 자동화된 테스트 시나리오 생성 방법을 설명한다. 마지막으로 테스트 시나리오의 자동 실행 및 결과 생성 방법을 설명한다.

3.2.1 단위 객체 추출

단위 객체 추출 방법은 3가지로 나누어 질 수 있다.

첫 번째 방법은 Windows SDK에서 제공되는 Hook 관련 라이브러리를 사용하는 방법으로써, 참고문헌 [5]에서 사용했던 방식이다. 이는 테스트 대상 어플리케이션을 실행시켜 Hook 관련 라이브러리를 이용하여 동적으로 생성된 단위 객체들의 정보를 획득하는 방법으로 해당 객체를 제어할 수 있는 핸들(HANDLE)과 ID 정보를 얻게 된다. 하지만 실제 생성된 단위 객체의 정보만 추출되기 때문에 최초 GUI MAP을 구성하기가 힘들다는 단점이 있다. 또한 명령객체가 다른 어느 객체에 영향을 주는지는 테스터가 직접 판단해야 한다.

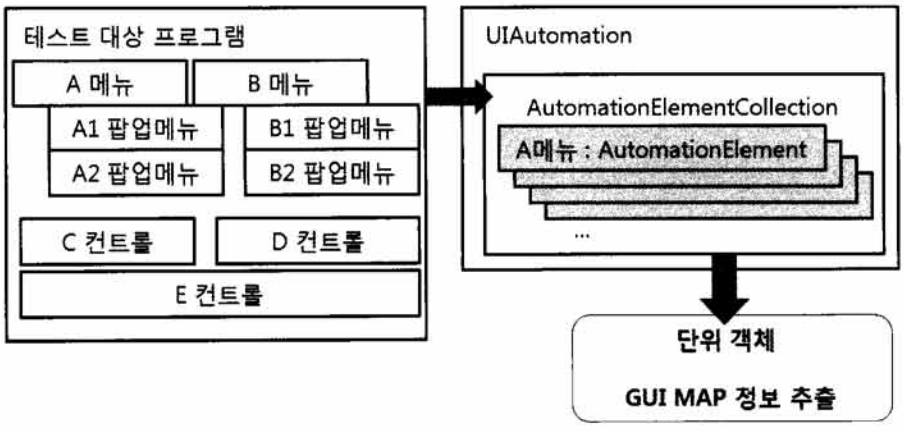
두 번째 방법은 PE Format(Portable Executable Format)과 MMF (Memory Mapping File) 기술을 사용하는 것으로써, 참고문헌 [9]에서 사용했던 방식이다. 이는 실행파일을 통해서 해당 프로그램이 가지고 있는 모든 컨트롤 및 메뉴의 정보를 획득할 수 있다. 실행파일에는 프로그램을 구성하는 모든 단위 객체에 대한 정보가 담겨있다. 이를 통해 GUI MAP을 구성하기가 훨씬 수월하다는 장점이 있다. 하지만 구현이 복잡하며, 표준 컨트롤 외에 사용자 정의 컨트롤이나, 다수의 컴포넌트들을 통해 프로그램이 구성되어 있다면 단위 객체를 가져오는 일이 더 복잡해지게 된다.

세 번째 방법은 이 논문에서 사용한 방식으로 .NET Framework 3.0이상 버전에서 지원하는 UI Automation 클래스를 사용하는 것이다. UIA는 보조 기술 제품과 자동화된 테스트 도구의 필요를 충족시키기 위해 개발되었다. UIA의 특징은 관리 코드로 Visual C# 또는 Visual Basic.NET을 사용하여 쉽게 프로그래밍 할 수 있다는 것이다. 그리고 사용자 인터페이스를 만들기 위해 새롭게 제시한 모델인 WPF(Windows Presentation Foundation) 요소를 지원하고 있다. UIA는 아래와 같은 구현상의 편리함을 제공한다.

- 객체를 AutomationElement로 표현함으로써 = 연산자 또는 Equals메서드를 사용하여 객체 간 런타임 식별이 쉽게 이루어진다.
- 탐색에 있어서도 TreeWalker 클래스를 사용하여 원하는 조건에 맞게 필터링 된 트리뷰(tree view)를 사용할 수 있으며 탐색 관련 메소드(FindFirst, FindAll 등)를 이용할 수 있어 효율적이다.
- 객체의 기능은 전용 인터페이스 구현을 통해서 공급자가 지원하는 컨트롤 패턴이 결정된다. 즉, 컨트롤 패턴을 결합함으로써 특정 UI요소가 지원하는 전체 기능 집합을 설명할 수 있다.
- 이벤트 핸들러를 사용할 수 있다.

이 논문에서는 .NET Framework 3.5 버전에서 지원하는 UIA 클래스를 사용하여 응용 프로그램을 분석하고 응용 프로그램이 소유한 단위 객체의 속성 정보를 추출한다. 기본 개념은 AutomationElementCollection 클래스를 이용하여 현재 생성되어 있는 모든 단위 객체(메뉴, 컨트롤)들을 추출한다. 그리고 AutomationElementCollection 클래스에 저장되어 있는 단위 객체를 각각 추출하게 되는데 이 때 추출되는 객체는 AutomationElement 클래스 타입이 된다. 단위 객체라 함은 AutomationElement 클래스의 객체를 의미하고 해당 클래스 객체는 단위 객체의 다양한 정보들을 담고 있다.

다음의 <표 2>는 AutomationElement의 속성들이다. 속성들 중에는 해당 요소가 어떠한 패턴의 요소인지를 확인할 수 있는 멤버가 있음을 알 수 있다. UIA에서는 이를 통해 요소의 패턴을 확인하여 필요시 해당 패턴의 속성을 얻어올 수 있다[11].



(그림 1) UIAutomation을 이용한 단위 객체 정보 추출

〈표 2〉 AutomationElement 속성들(패턴 관련)

속성	설명
IsExpandCollapsePatternAvailableProperty	ExpandCollapse 패턴인지 여부
IsGridItemPatternAvailableProperty	GridItem 패턴인지 여부
IsGridPatternAvailableProperty	Grid 패턴인지 여부
IsInvokePatternAvailableProperty	Invoke패턴인지 여부
IsMultipleViewPatternAvailableProperty	MultipleView 패턴인지 여부
IsRangeValuePatternAvailableProperty	RangeValue 패턴인지 여부
IsScrollItemPatternAvailableProperty	ScrollItem 패턴인지 여부
IsScrollPatternAvailableProperty	Scroll 패턴인지 여부
IsSelectionItemPatternAvailableProperty	SelectionItem 패턴인지 여부
IsSelectionPatternAvailableProperty	Selection 패턴인지 여부
IsTableItemPatternAvailableProperty	TableItem 패턴인지 여부
IsTablePatternAvailableProperty	Table 패턴인지 여부
IsTextPatternAvailableProperty	Text 패턴인지 여부
IsTogglePatternAvailableProperty	Toggle 패턴인지 여부
IsTransformPatternAvailableProperty	Transform 패턴인지 여부
IsValuePatternAvailableProperty	Value 패턴인지 여부
IsWindowPatternAvailableProperty	Window 패턴인지 여부

〈표 3〉 UIA의 컨트롤 패턴

패턴	설명	예
ExpandCollapsePattern	확장하거나 축소 가능한 컨트롤	메뉴 항목
GridPattern	표 기능을 지원하는 컨트롤	표
GridItemPattern	표의 개별 셀	표 안의 셀
InvokePattern	호출할 수 있는 컨트롤	버튼
MultipleViewPattern	여러 형태로 보여줄 수 있는 컨트롤	목록 뷰
RangeValuePattern	적용할 값의 범위가 있는 컨트롤	업다운 컨트롤
ScrollPattern	표시 내용이 많을 경우 스크롤 바 포함 컨트롤	에디트
ScrollItemPattern	스크롤 되는 목록에서 개별 항목	리스트 항목
SelectionPattern	선택 가능한 컨트롤	콤보 박스
SelectionItemPattern	선택 가능한 컨트롤의 아이템	리스트 항목
TablePattern	테이블 컨트롤	워크시트
TableItemPattern	테이블의 항목	워크시트 셀
TextPattern	문서를 보여주거나 편집 가능한 컨트롤	에디트
TogglePattern	상태가 전환되는 컨트롤	체크 박스
TransformPattern	크기 및 위치 등이 조절되는 컨트롤	
ValuePattern	값을 지정하거나 얻어올 수 있는 컨트롤	에디트
WindowPattern	MS의 기본적인 개념에서의 창 컨트롤 양식의 맨 아래	대화 상자

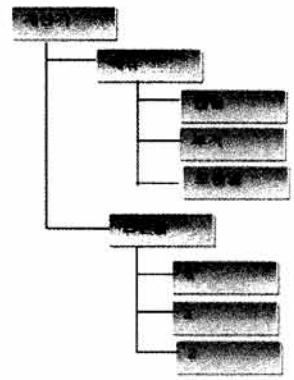
다음 <표 3>에는 UIA의 컨트롤 패턴들에 대한 설명과 예가 있다. 컨트롤에 따라 패턴은 다르게 구성되어 있으며 이는 테스트 과정 중 Recording 후 Play시 사용된다.

3.2.2 GUI MAP 구성

.NET Framework에서 제공하는 UIA를 사용하여 단위 객체를 추출하여 GUI MAP을 구성한다. GUI MAP은 테스트 대상 프로그램이 가지고 있는 단위 객체들을 트리구조 형태로 표현한다.

GUI MAP 구성 과정이 필요한 이유는 다음과 같다. 첫 번째로 GUI MAP을 구성함으로써 이벤트 흐름을 테스터에게 보여준다. 테스터는 여러 가지 문서들을 통해서 프로그램의 흐름을 이해하겠지만, 단위 객체들의 호출흐름을 트리구조 형태로 제공함으로써 좀 더 쉽게 이벤트 흐름을 파악할 수 있다.

두 번째로 테스트 수행 후 커버리지를 측정하는 기준이

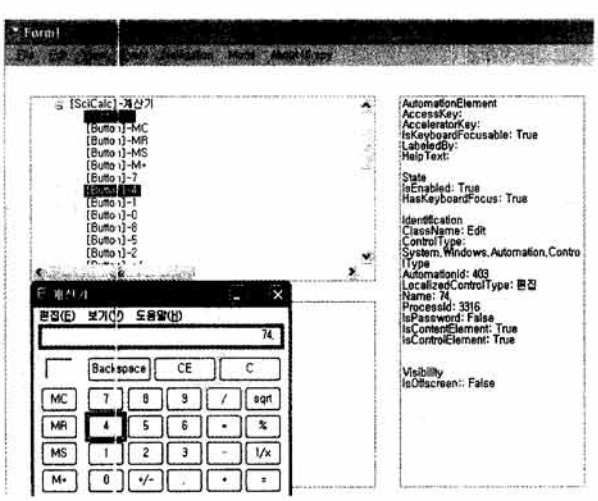
XML	트리형태 표현
<계산기> <메뉴> <메뉴아이템>편집</메뉴아이템> <메뉴아이템>보기</메뉴아이템> <메뉴아이템>도움말</메뉴아이템> </메뉴> <컨트롤> <버튼>0</버튼> <버튼>1</버튼> <버튼>2</버튼> </컨트롤> </계산기>	

(그림 2) GUI MAP의 예

된다. 여기서 커버리지는 대상 프로그램에서 발생할 수 있는 전체 이벤트 흐름 중 실제 테스트 된 이벤트 흐름의 비율을 의미한다. GUI 프로그램은 그 규모에 따라 수많은 이벤트 흐름들이 발생할 수 있고, 수많은 이벤트 조합이 발생할 수 있다. 그것들을 모두 확인해 가면서 테스트 한다는 것은 쉬운 일이 아니다. GUI MAP은 프로그램 구동에 필요한 전체 이벤트 흐름을 가지고 있기 때문에 다양한 시나리오로 테스트를 수행했을 경우 커버리지 측정을 자동으로 할 수 있게 된다.

세 번째로 단위 객체 그룹화의 초기과정이다. 테스터에 의해 기본 시나리오(의미 있는 다양한 시나리오를 생성하기 위한 최초 시나리오)가 생성되게 되면 단위 객체별 그룹화를 수행하게 되는데, 이 때 GUI MAP에 등록된 객체들의 정보를 가지고 그룹화를 수행한다.

GUI MAP 구성은 .NET Framework 3.0 이나 3.5 기반에서 제공하는 UIA를 통해 단위 객체를 추출해 내고, 추출된 객체들을 실행시키면서 구조의 일부를 자동으로 파악하게 된다. 완벽한 구조는 테스터의 추가적인 보정작업을 거쳐 얻게 된다.



(그림 3) UI Automation을 이용한 단위 객체 추출

(그림 3)은 UI Automation을 이용한 단위 객체 추출 사례이다. 좌측에 보이는 뷰어는 계산기의 GUI MAP 부분이며, 우측에 보이는 뷰어는 단위 객체의 속성 정보이다. 좌측 뷰어에서 하나의 객체를 선택하게 되면 우측에 해당 객체의 속성 정보를 출력한다.

3.2.3 단위 객체 그룹화 및 입력 데이터 설정

GUI MAP을 통해 의미 있는 다수의 시나리오를 생성하기 위해서는 단위 객체의 특성에 맞추어 단위 객체를 분류하여 그룹화를 해야 한다. 그룹화는 동일한 객체라도 프로그램에 따라서 사용 패턴이 다르기 때문에 일반화하기 어렵다. 따라서 그룹화는 테스터에 의해 수행되게 된다. 테스터는 작성된 하나의 시나리오를 수행하게 되며, 단위 객체별 그룹화를 수행하게 된다. 이는 다음과 같이 진행된다.

① 기본 시나리오 생성

테스터는 응용 프로그램의 명세를 참조하여 응용 프로그램을 작동한다. 이 때 테스터가

작동한 이벤트를 저장하여 기본 시나리오로 이용한다. 예를 들어, 계산기의 경우 '버튼 1', '버튼+', '버튼2', '버튼='의 순서로 작동했을 경우 이 수행 순서로 "1+2 = " 와 같은 기본 시나리오가 기록된다.

② 단위 객체 그룹화 수행

그룹화는 기본 시나리오를 확장하기 위한 사전단계이며, 의미 있는 다양한 시나리오를 생성하기 위한 과정이다. 예를 들어 만약 "3+5 = "와 같은 기본 시나리오가 존재한다면, '3'과 '5'는 '0'~'9' 의 객체로, '+'는 '+', '-', '*', '/' 사이의 객체로 그룹화를 수행하게 된다.

③ 기본 시나리오의 특성상 그룹화가 필요한 경우가 있고, 그룹화가 필요하지 않는 경우도 있다. 그룹화가 필요 없더라도 단위 객체가 실행 될 때 다양한 데이터를 변경해 가면서 수행되어야 할 경우도 있다. 이를 위해 변경 가능한 입력데이터를 그룹화 과정에서 별도 적용함으로써, 다양한 시나리오 생성을 가능하게 한다.

〈표 4〉 그룹화 예

내용	입력 이벤트	프로그램
시나리오	[버튼:3] - [버튼:+] - [버튼:5] - [버튼:=]	계산기

내용	인덱스	그룹화 객체	비고
그룹화	G1	버튼0, 버튼1, 버튼2, 버튼3, 버튼4, 버튼5, 버튼6, 버튼7, 버튼8, 버튼9	정보객체
	G2	버튼+, 버튼-, 버튼*, 버튼/	명령객체

내용	시나리오	그룹화 인덱스
시나리오와 그룹 연결	버튼3	G1
	버튼+	G2
	버튼5	G1
	버튼=	-

3.2.4 테스트 시나리오 생성

GUI 컴포넌트 테스트에서는 다양하고 많은 테스트가 수행되어 GUI 컴포넌트의 안전성, 견고성, 사용성을 향상시켜야 한다. 일반적으로 테스터는 테스트 시나리오를 결정하고 그 시나리오를 바탕으로 테스트 데이터를 변경하여 GUI 테스트를 수행한다. 이와 같은 방법은 하나의 시나리오에서 데이터 값만을 변경하는 효과를 갖게 된다. 그러나 우리는 명령 객체를 교체하고 정보 객체를 변경함으로써 보다 다양한 시나리오를 생성하고자 한다.

여기서 시나리오의 의미는 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.

1) 시나리오 생성

3.2.3 절에서 설명된 단위 객체 그룹화를 기반으로 의미 있는 다양한 시나리오를 생성하는 과정이다. 그룹별 조합 혹은 입력 데이터를 변경함으로써 다양한 테스트 시나리오를 생성한다. 시나리오 생성 방법은 다음과 같다.

① 기본 시나리오 생성 및 추상화

테스터는 응용 프로그램의 명세를 참조하여 기본 시나리오를 명시적으로 작성한다. 그리고 그룹화에서 정의된 그룹 인덱스를 기반으로 시나리오를 추상적으로 표현한다.

② 그룹화된 객체 변경을 통한 시나리오 생성

그룹 정보를 기반으로 그룹 내의 단위 객체를 다른 객체로 대체하여 새로운 시나리오를 생성한다.

③ 입력 데이터 변경을 통한 시나리오 생성

입력 데이터가 설정 되어 있는 경우 입력 데이터를 변경하여 새로운 시나리오를 생성한다.

④ 생성된 시나리오 검증

②번 과정에서 생성된 시나리오가 해당 그룹의 모든 객체들을 포함하고 있는지 검증한다.

③번 과정에서 생성된 시나리오가 변경 가능한 데이터를 모두 포함하고 있는지 검증한다.

⑤ 검증 결과에 따라 추가적인 시나리오를 생성한다.

위와 같이 객체 변경 및 데이터 변경을 통해서 다양한 시나리오를 생성하게 된다. 이는 기존 테스트 도구에서 수동으로 처리되었던 객체 변경 및 데이터 변경 부분을 자동화 했다는 점에서 의미가 있다.

3.2.5 테스트 수행 및 결과물 생성

3.2.4절에서 생성된 다양한 시나리오들을 가지고 테스트하면서 올바로 수행되었는지 여부를 자동으로 판단하는 것은 매우 어려운 일이다. 명령객체를 통해 진행되어지는 결과가 정상인지 비정상인지 판단할 수 있는 기준이 없기 때문이다. 물론 사전에 비교 데이터나 이미지가 주어진 경우에 한해서만 자동화는 가능할 수 있을 것이다. 이런 이유로 이 논문에서는 결과 판단의 자동화에 대한 부분은 고려하고 있지 않다.

하지만 테스터에게 테스트 실행의 결과물은 이미지 형태나 데이터 형태로 보여준다. 이를 통해 테스트 결과를 분석하는데 도움을 주고자 함이다. 또한 테스트 과정에서 오류가 발견되더라도 타이밍의 부적절 성으로 인해 재현하지 못하는 경우가 발생하게 되는데 이 때 이미지 형태로 근거를 제시하면 테스터에게 도움이 될 것이다.

테스트 실행의 결과물의 제공 내용은 다음과 같다. 첫 번째로 모든 테스트 시나리오별 단위 객체 실행 이미지를 저장하여 제공한다. 이를 통해 테스터는 실제 모든 시나리오를 직접 테스트하지 않아도 이미지를 통해 간접적으로 결과물을 점검하게 된다. 두 번째로 커버리지 결과를 제공한다. 커버리지 내용은 다음과 같다.

- 커버리지 결과 : 프로그램에서 발생할 수 있는 이벤트 전체 흐름과 실제 테스트 된 이벤트 흐름 비율
- 테스트 수행 횟수 : 특정 이벤트 흐름의 실제 테스트 수행 횟수

이를 통해 미처 테스트 하지 못했던 부분이나, 테스트 수행 횟수가 부족한 부분들을 다시 테스트 할 수 있도록 해준다.

GUI MAP	테스트 시나리오 경로		
	인덱스	경로	수행 횟수
	R1	[메뉴]-[편집]	2
	R2	[메뉴]-[보기]	5
	R3	[메뉴]-[도움말]	0
	R4	[컨트롤]-[G1]	20

(그림 4) 커버리지 측정 예

커버리지는 다음과 같이 측정한다. 테스트 수행 전 GUI MAP은 해당 프로그램에서 발생할 수 있는 모든 이벤트 시나리오를 가지고 있다. 실제 테스트를 수행하게 되면 테스트 수행 경로를 GUI MAP과 비교하여 어떠한 경로가 테스트되었는지 여부를 판단한다. 이를 통해 특정 경로가 몇 번 테스트 되었는지, 혹은 어떤 경로는 테스트 되지 않았는지를 판단할 수 있다.

(그림 4)는 커버리지 측정 사례를 보여준다. 좌측의 GUI MAP에 의하면 6가지 경로가 존재한다. 하지만 우측의 테스트 시나리오 경로는 4가지만 설정되어 있다. 그 이유는 컨트롤 0, 1, 2의 경우 같은 그룹이기 때문에 동일한 경로로 표현한 것이다. 따라서 R4의 시나리오 경로 표현 시 그룹명인 G1이 사용된 것을 볼 수 있다. 수행 횟수는 실제 테스트가 수행된 횟수이다. R3의 경로는 한 번도 테스트가 수행되지 않았음을 알 수 있고, R4는 가장 많은 테스트가 수행되었음을 알 수 있다.

4. 사례연구 : 간단한 회원관리 프로그램 테스트

이 절에서는 논문에서 제시한 방식으로 구현된 GUI 테스트 도구를 이용하여 테스트를 수행한 내용과 그 결과에 대해 기술한다. (그림 5)는 테스트 대상 프로그램이다. 총 3개의 윈도우와 메뉴로 구성되어 있는데 상세 내용은 <표 5> 테스트 프로그램 명세에 자세하게 기술되어 있다.

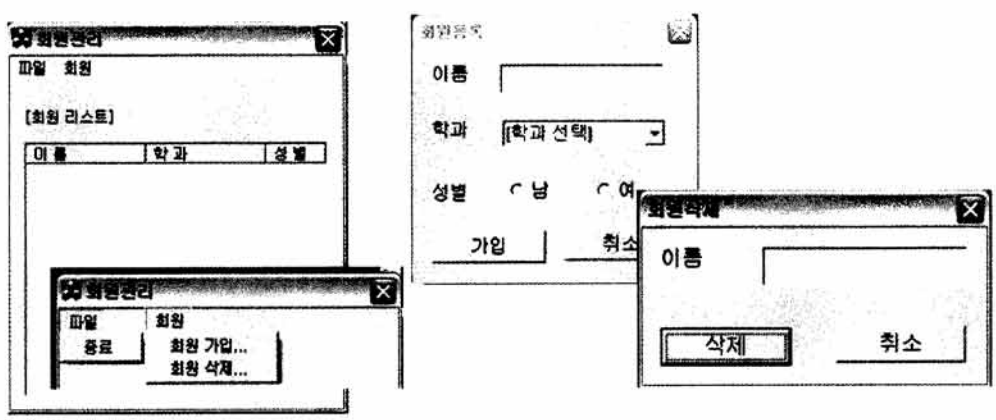
4.1 테스트 시나리오 자동 생성

먼저 UIA를 이용하여 자동으로 GUI MAP을 구성한다. 그 후 테스터에 의해 하나의 테스트 시나리오가 생성되며, 그룹화 및 입력 데이터 설정 작업을 수행하게 된다. 아래 그림은 XML로 기록된 하나의 테스트 시나리오에 따른 그룹화 및 입력 데이터 설정 작업 사례이다.

(그림 6)의 XML 파일 구성은 다음과 같다.

- <Item> : 해당 윈도우의 단위 객체
- <Scenario> : 실제 테스터가 입력한 시나리오 정보
- <DataGroup> : 입력된 데이터
- <Group> : 컨트롤 그룹

Scenario 태그 내에 저장된 단위 객체들은 GroupID 값을 가지고 있다. 이는 <DataGroup>이나 <Group>에 등록되어 있는 그룹들의 인덱스이다. 시나리오 생성 시 해당 정보를 기반으로 다양한 조합을 생성하게 된다.



(그림 5) 테스트 프로그램

〈표 5〉 테스트 프로그램 명세

• 메인 윈도우 : 회원 관리
메뉴를 구성하고 있으며 메인윈도우에 현재 등록된 회원 리스트를 출력한다.

객 체		내 용
컨트롤	정적	"[회원 리스트]" 문자열 출력
	리스트뷰	등록된 회원 리스트를 보여준다. 회원 정보는 이름, 학과, 성별이다.
메뉴	파일	메뉴바를 구성한다.
	회원	메뉴바를 구성한다.
	종료	프로그램을 종료한다.
	회원 가입....	회원 등록 윈도우를 생성한다.
	회원 삭제...	회원 삭제 윈도우를 생성한다.

• 팝업 윈도우 : 회원 등록
회원 등록 정보를 입력받아 저장한다.

객체		내용
컨트롤	정적	"이름" 문자열 출력
	에디트	이름을 입력 받는다.
	정적	"학과" 문자열 출력
	콤보박스	학과 선택을 요청한다. 선택 데이터는 "컴퓨터학과", "IT학과", "게임학과" 이다.
	정적	"성별" 문자열 출력
	라디오버튼	"남"을 선택한다.
	라디오버튼	"여"를 선택한다.
	버튼	입력 정보를 저장한다.
	버튼	입력 정보를 저장하지 않는다.

• 팝업 윈도우 : 회원 삭제
회원 이름을 입력받아 해당 정보를 삭제한다.

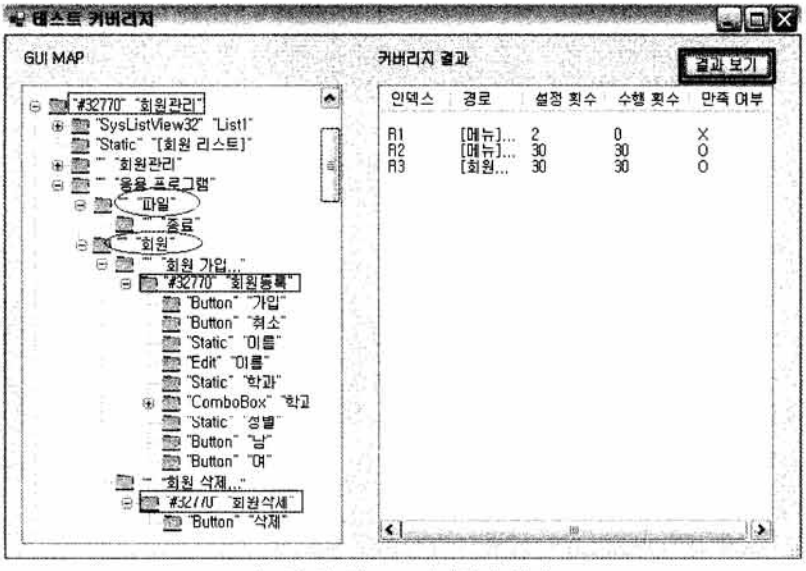
객체		내용
컨트롤	정적	"이름" 문자열 출력
	에디트	삭제할 이름을 입력 받는다.
	버튼	검색된 정보를 삭제한다.
	버튼	삭제를 취소한다.

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
- <ScenarioGroupData>
  - <Item>
      <Identification ProcName="회원등록" Control="이름" ClassName="edit" AutomationID="250" />
      <Identification ProcName="회원등록" Control="학과" ClassName="combobox" AutomationID="180" />
      <Identification ProcName="회원등록" Control="남" ClassName="radiobutton" AutomationID="120" />
      <Identification ProcName="회원등록" Control="여" ClassName="radiobutton" AutomationID="132" />
      <Identification ProcName="회원등록" Control="가입" ClassName="button" AutomationID="122" />
      <Identification ProcName="회원등록" Control="취소" ClassName="button" AutomationID="128" />
    </Item>
  - <Scenario>
      <Identification ProcName="회원등록" Control="이름" ClassName="edit" AutomationID="250" GroupID="D1" />
      <Identification ProcName="회원등록" Control="학과" ClassName="combobox" AutomationID="180" GroupID="D2" />
      <Identification ProcName="회원등록" Control="남" ClassName="radiobutton" AutomationID="120" GroupID="G1" />
      <Identification ProcName="회원등록" Control="가입" ClassName="button" AutomationID="122" GroupID="G2" />
    </Scenario>
  - <DataGroup>
      <Identification ProcName="D1" Data="김길동" />
      <Identification ProcName="D1" Data="고길동" />
      <Identification ProcName="D1" Data="강길동" />
      <Identification ProcName="D1" Data="최길동" />
      <Identification ProcName="D1" Data="홍길동" />
      <Identification ProcName="D2" Data="컴퓨터학과" />
      <Identification ProcName="D2" Data="IT학과" />
      <Identification ProcName="D2" Data="게임학과" />
    </DataGroup>
  - <Group>
      <Identification ProcName="G1" AutomationID="120" />
      <Identification ProcName="G1" AutomationID="132" />
      <Identification ProcName="G2" Data="122" />
      <Identification ProcName="G2" Data="128" />
    </Group>
</ScenarioGroupData>
```

(그림 6) 그룹화/입력 데이터의 XML 저장

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
- <Scenario>
  - <Instance>
      <Identification ProcName="회원등록" Control="이름" ClassName="edit" AutomationID="250" Data="김길동" />
      <Identification ProcName="회원등록" Control="학과" ClassName="combobox" AutomationID="180" Data="IT학과" />
      <Identification ProcName="회원등록" Control="남" ClassName="radiobutton" AutomationID="120" />
      <Identification ProcName="회원등록" Control="가입" ClassName="button" AutomationID="122" />
    </Instance>
  - <Instance>
      <Identification ProcName="회원등록" Control="이름" ClassName="edit" AutomationID="250" Data="최길동" />
      <Identification ProcName="회원등록" Control="학과" ClassName="combobox" AutomationID="180" Data="IT학과" />
      <Identification ProcName="회원등록" Control="남" ClassName="radiobutton" AutomationID="120" />
      <Identification ProcName="회원등록" Control="가입" ClassName="button" AutomationID="122" />
    </Instance>
  - <Instance>
      <Identification ProcName="회원등록" Control="이름" ClassName="edit" AutomationID="250" Data="김길동" />
      <Identification ProcName="회원등록" Control="학과" ClassName="combobox" AutomationID="180" Data="컴퓨터학과" />
      <Identification ProcName="회원등록" Control="여" ClassName="radiobutton" AutomationID="132" />
      <Identification ProcName="회원등록" Control="취소" ClassName="button" AutomationID="128" />
    </Instance>
  - <Instance>
      <Identification ProcName="회원등록" Control="이름" ClassName="edit" AutomationID="250" Data="홍길동" />
      <Identification ProcName="회원등록" Control="학과" ClassName="combobox" AutomationID="180" Data="IT학과" />
      <Identification ProcName="회원등록" Control="남" ClassName="radiobutton" AutomationID="120" />
      <Identification ProcName="회원등록" Control="취소" ClassName="button" AutomationID="128" />
    </Instance>
  - <Instance>
      <Identification ProcName="회원등록" Control="이름" ClassName="edit" AutomationID="250" Data="강길동" />
      <Identification ProcName="회원등록" Control="학과" ClassName="combobox" AutomationID="180" Data="게임학과" />
      <Identification ProcName="회원등록" Control="여" ClassName="radiobutton" AutomationID="132" />
      <Identification ProcName="회원등록" Control="가입" ClassName="button" AutomationID="122" />
    </Instance>
</Scenario>
```

(그림 7) 테스트 시나리오 확장 XML 파일



(그림 8) 테스트 커버리지 결과

4.2 테스트 시나리오 확장

4.1에서 저장된 XML 파일을 소스로 하고 생성 개수를 설정하여 시나리오를 자동으로 생성한다. 이 때 프로그램은 자체적으로 그룹화 되어있는 단위 객체가 어느 정도 사용되는지 여부를 출력해 주게 되고, 이를 통해 추가적으로 시나리오를 추가할지 여부를 테스터가 판단한다. (그림 7)은 XML 파일에 저장된 확장 시나리오이다. 확장 시나리오의 단위 객체의 데이터 부분과 라디오버튼 부분의 변경을 통해 확장된 시나리오를 볼 수 있다.

4.3 테스트 수행 및 결과

테스트 결과물은 다음과 같다.

• XML 파일 형식으로 저장되어 있는 시나리오

실제 테스트된 시나리오를 말하며 (그림 7)의 내용과 동일하다.

• 테스트 저장 이미지

이벤트 시나리오에 의해 테스트가 진행되면 객체 단위로 이벤트가 발생할 때마다 저장 된다. 저장은 대상 프로그램의 모든 윈도우를 개별 저장하게 된다.

• 테스트 커버리지

테스트 가능한 전체 경로와 실제 테스트 된 경로 및 횟수를 측정해 제공한다. 이를 통해 테스터는 추가적인 테스트 수행 여부를 결정하게 된다. (그림 8)은 테스트 커버리지 측정 사례를 보여주고 있다.

(그림 8)의 오른쪽 창에서 인덱스는 이벤트 시나리오를 구분하는 기본 값을 의미하고, 경로는 실제 발생 가능한 이벤트 흐름을 나타낸다. 설정 회수는 테스트 하고자 하는 회수를 나타내고 수행 회수는 실제 테스트가 수행된 회수이다. 끝으로 만족 여부는 설정 회수와 수행 회수를 비교하여 설정 회수만큼 테스트가 실제로 실행되었는지 여부를 나타낸다.

인덱스 R1의 경로는 [메뉴:파일]-[메뉴:종료]이고, 테스터가 요구하는 테스트 횟수는 2번이지만, 실제 수행횟수는 0이므로 테스트 커버리지를 만족하지 못하였음을 알 수 있다. R2 는 [메뉴:회원]-[메뉴:회원가입]을 통해서 (그림 5)와 같은 회원등록 팝업 윈도우를 생성시키고, 이름, 학과, 성별을 설정한 후 가입버튼을 선택한 경로이며, R3는 [메뉴:회원]-[메뉴:회원삭제]를 통해서 회원삭제 팝업 윈도우를 생성시키고, 이름을 입력한 후 삭제버튼을 선택한 경로이다.

5. 결론 및 토의

GUI 기반의 응용 소프트웨어는 외부 이벤트를 처리할 수 있는 메뉴 및 다양한 컨트롤을 포함하고 있으며, 이를 통해 사용자는 소프트웨어를 구동하게 된다. 따라서 사용자의 이벤트를 받아들이는 GUI 객체의 검증은 반드시 필요하다. 하지만 GUI 컴포넌트를 테스트하는 것은 테스터에게 많은 노력을 요구한다. 이러한 GUI 테스트의 어려움들을 해결하고자 많은 연구와 테스트 자동화 도구들이 개발되었다.

기존 GUI 테스트 도구의 테스트 케이스 생성 방법은 테스터에게 많은 개입을 요구하였다. 이 논문은 이러한 테스터의 개입을 줄여주는 GUI 자동화 방법을 제시하였다. 이 방법은 GUI의 흐름을 자동으로 분석하여 GUI MAP을 구성한다. 그리고 단위 객체를 그룹화하고 하나의 테스트 시나리오에서 같은 그룹 안에 단위 객체들을 교체하거나, 입력 정보의 값을 변경함으로써 다양하고 효율적인 테스트 시나리오를 생성하는 방법을 제안 하였다. 또한 기존 GUI 테스트 도구는 테스트 시나리오의 커버리지를 제시하지 못하였으나 이 논문에서는 GUI MAP의 구성을 통해 발생할 수 있는 모든 시나리오를 분석할 수 있기 때문에 실제 테스트 수행 시 어떠한 부분이 테스트 되었는지, 어떠한 부분이 테스트 되지 않았는지를 자동으로 판단할 수 있어서 커버리지를 측정할 수 있다.

구현을 용이하게 하기위해 사용한 .NET 프레임워크의 UIA는 단위 객체 추출과정, 이벤트 저장 및 실행 부분에 사용되었고, 이는 기존 그룹화 방식을 이용해 테스트 도구 제작에 사용되었던 API를 이용한 방식보다 구현의 편리성을 제공하였다.

앞선 연구에서 자동화 테스트와 비교하여 매뉴얼 테스트가 두 배 이상의 시간과 비용을 요구한다는 사실을 제시하고 있다[8]. 물론 "어떠한 방식으로 자동화 할 것인가?" "어떠한 도구를 사용하는가?"에 따라 시간과 비용의 차이가 발생하겠지만, 자동화를 통해 얻는 테스터의 부하 감소 및 정량적 개선점이 발생한다는 점은 분명하다. 또한 참고문헌 [5]에서는 테스터가 직접 테스트 시나리오를 작성할 때 소요되는 시간과 그룹화 테스트 기법을 사용하여 테스트 시나리오를 생성할 때 소요되는 시간에 대한 정량적인 비교 결과가 제시되어 있다. 이 논문은 참고문헌 [5]와 [9]에서 제시된 그룹화 테스트 기법을 더욱 개선하는 방법을 제시하는 논문이므로 정량적인 비교보다는 이 연구가 기존 연구의 어떤 측면을 개선하였는가를 기능적인 측면에서 비교한다. 이 논문에서 제시한 도구와 기존 도구와의 기능적 차별성을 정리하면 다음과 같다.

첫 번째로 테스트 스크립트 생성시 Record-Playback 방식과 본 논문 제시 방법은 이벤트를 저장함으로써 자동화한다. 하지만 나머지 방식은 GUI 컨트롤을 통해 일부 자동화 하였거나, 자동화 방식을 제시하지 못하였다.

두 번째로 그룹화 방식을 통해서 시나리오의 자동 확장 방법을 본 논문과 그룹화 방식에서는 제시하였지만 기존 Record-Playback 방식과 명세 기반 방식은 수동적인 입력을 통해 시나리오를 확장해 나간다.

〈표 6〉 테스트 기법 비교

비교 내용	Record-Playback	명세 기반	그룹화	이 논문 제시 기법
테스트 스크립트 생성 자동화	O	△	O	O
시나리오 자동 확장	X	X	O	O
멀티 윈도우 환경 테스트	O	O	X	O
테스트 수행 자동화	O	O	O	O
테스트 커버리지 측정	X	X	X	O
체크 포인트	X	X	O	△
UI 컨트롤 입력 데이터 값 변경 자동화	△	△	X	O
테스트 실행을 위한 프로그램 수행 환경 지원	O	O	X	X

1) Record-Playback은 Ranorex Recorder 도구를 기반으로 함
2) 명세 기반 기법은 TestComplete 6 도구를 기반으로 함
3) 그룹화는 참고문헌 [5]에서 제시된 도구를 기반으로 함

세 번째로 GUI 프로그램은 다양한 윈도우로 구성되어 있는데 그룹화 방식은 이러한 환경에서의 테스트 수행이 불가능하다. 하지만 나머지 방식은 가능하며 이 논문에서도 GUI MAP 구성을 통해 다양한 윈도우로 구성된 환경에서도 가능하다.

네 번째로 그룹화 방식을 제외한 나머지 방식은 생성된 테스트 시나리오를 자동으로 수행한다.

다섯 번째로 테스트 커버리지는 본 논문에서 제시한 부분이다. 이를 위해서 GUI MAP을 구성하였고, 테스트 수행 시 GUI MAP과 비교하여 커버리지 여부를 판단하였다.

여섯 번째는 체크 포인트이다. 체크 포인트란 테스터가 관심을 가지는 부분을 체크하여 테스트 수행 후 그 결과를 확인 할 수 있는 방법을 제시하는 부분인데, 그룹화 방식에서는 체크된 단위 객체 수행 결과 이미지를 저장하여 보여준다. 하지만 이 논문의 제시 기법에서는 모든 단위 객체에 대한 이미지를 저장하기 때문에 체크 포인트 방식을 제시하지 않았지만, 해당 기능을 다른 기능으로 대체하였다고 볼 수 있다.

일곱 번째로 UI 컨트롤 입력 데이터 값 변경 자동화이다. 에디트 컨트롤의 경우 데이터 입력을 필요로 한다. Record-Playback 방식과 명세 기반 방식은 저장된 모든 스크립트에서 데이터 입력 부분을 수정함으로써 일부 자동화를 수행했지만, 이 논문의 방법에서는 단위 객체에 변경 데이터들을 미리 설정하여 저장함으로써 데이터 변경을 통한 시나리오 생성을 자동화 하였다.

마지막은 테스트 실행을 위한 프로그램 수행 환경 지원 부분이다. 이는 저장된 시나리오를 프로그램 언어로 코드화해서 테스터에게 제시하는 기능이며, 본 논문에서는 GUI 테스트의 주기능이 아닌 부가적인 기능이라고 판단해서 해당 기능을 고려하지 않았다.

이 논문에서 제안한 테스트 시나리오 생성 방법으로 다음과 같이 GUI 컴포넌트 테스트의 어려움을 개선할 수 있다.

첫 번째로 GUI 테스트의 어려움 중 테스터에게 큰 부담이 되었던 시나리오 생성 과정에서 GUI 객체들을 임의로 조합하는 대신 그룹화 된 객체들을 선택적으로 조합함으로써 다양하고 의미 있는 테스트 시나리오를 대량으로 생성하여 테스터의 부담을 줄였다.

두 번째로 시나리오 상의 단위 객체 중 입력데이터를 설정 할 수 있는 객체일 경우 테스터에 의해 입력 데이터를 설정하게 하여 입력 데이터의 변동에 의한 테스트 시나리오를 생성하였다. 이는 테스터가 스크립트 상에서 입력 데이터를 직접 수정하여 시나리오를 생성하는 부담을 줄이게 한다.

세 번째로 생성된 다수의 시나리오를 자동으로 테스트 하고, 테스터는 테스트가 종료된 후에 저장된 이미지를 통해 오류 여부를 검증할 수 있게 하였다.

네 번째로 수행해야 할 GUI 테스트 범위를 자동으로 인식함으로 커버리지를 측정할 수 있을 뿐만 아니라 실제 수행되지 않은 부분들을 체크할 수 있게 하였다.

그러나 이 논문에서 제안한 방법 및 도구는 몇 가지 보완해야 할 문제점이 있다.

첫 번째, 이 논문에서 제시하는 GUI 테스트 자동화 기법은 테스트 계획, 설계, 실행, 평가라는 테스팅 프로세스의 각 단계에서 자동화 대상이 되는 설계, 실행, 평가 단계에서 적절한 기법을 제시함으로써 어떠한 GUI 환경에도 적용 가능한 테스트 방법이다. 다만 테스트 케이스 생성 과정을 지원하기 위하여 .NetFramework 3.5 기반의 UIAutomation 클래스 라이브러리를 이용하여 제작되었기 때문에 플랫폼에 종속적이다. 그렇지만 이 부분은 3.2.1절에서 언급된 단위 객체 추출의 방법 중 다른 방법을 선택한다면 하부 플랫폼에 종속적인 문제를 해결할 수 있다.

두 번째, 테스트 이후 GUI 컨트롤 등 UI 변경이 발생할 수 있다. 만약 특정 컨트롤의 좌표 변경이라면 논문에서 제시된 방법을 통해 유연하게 대처할 수 있지만 컨트롤이 삭제되거나 추가될 경우 기존 테스트 시나리오들 중 일부는 필요 없게 될 것이다. 이러한 부분을 고려한 기존 테스트 시나리오의 효율적 관리 방안에 대한 연구가 필요하다.

세 번째로, 테스트 시나리오의 커버리지에 대한 깊이 있는 연구도 필요하다. 현재는 단순히 테스트 된 결과만을 가지고 측정하지만 좀 더 상세한 정보를 활용하여 결과를 제시하는 연구가 필요하다. 향후 이러한 문제점들을 개선하기 위한 연구를 계속 진행할 계획이다.

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. Sofa, "Hierarchical GUI Test Case Generation Using Automated Planning." IEEE Transactions on Software Engineering, Vol.27, No.2, pp.144-1445, Feb., 2001.

[3] Automated GUI testing, Tessella Support Services plc, January, 1999.

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

[5] 이정규, 국승학, 김현수, "시나리오의 자동 생성을 통한 GUI 테스트 케이스 생성 방법", 한국정보과학회논문지, 제36권 제1호, pp.45-53, 2009.

[6] Testcomplete 6, AutomatedQA.

[7] Ranorex, www.ranorex.com

[8] M. Sowers, Software Testing Tools Summary, Software Development Technologies Inc. White Paper, 2002.

[9] 최창민, 김현수, 국승학, "멀티윈도우 기반에서 시나리오 자동 생성을 통한 GUI 테스트 케이스 생성", 한국정보과학회 2009 한국컴퓨터종합학술대회 논문집, 제36권 제1호(B), pp23-28, 2009.6.

[10] 이정규, "시나리오 확장을 통한 GUI 테스트 케이스 자동 생성 도구의 설계 및 구현", 석사 학위 논문, 충남대학교, 2009.

[11] http://msdn.microsoft.com/ko-kr/library/ms788733(VS.90).aspx

최 창 민

e-mail : choiccmm@hanmail.net
2000년 건국대학교 낙농학과(학사)
2010년 충남대학교 컴퓨터공학과(공학석사)
2011년~현 재 충남대학교 컴퓨터공학과 박사과정
관심분야 : GUI 테스팅, 소프트웨어공학, 영상처리 등

정 인 상

e-mail : insang@hansung.ac.kr
1987년 서울대학교 컴퓨터공학과(학사)
1989년 한국과학기술원(KAIST) 전산학과 (공학석사)
1993년 한국과학기술원(KAIST) 전산학과 (공학박사)
1999년~현 재 한성대학교 컴퓨터공학과 교수
관심분야 : 소프트웨어 공학, 소프트웨어 테스팅

김 현 수

e-mail : hskim401@cnu.ac.kr
1988년 서울대학교 계산통계학과(학사)
1991년 한국과학기술원(KAIST) 전산학과 (공학석사)
1995년 한국과학기술원(KAIST) 전산학과 (공학박사)
1995년~1995년 한국전자통신연구원 Post Doc.
1996년~2001년 금오공과대학교 조교수
2001년~현 재 충남대학교 컴퓨터공학과 교수
관심분야 : 소프트웨어 공학, 소프트웨어 테스팅, SOA