소스코드기반의 GUI 테스트 자동화 기법의 구현
(An Automated Test Technique of GUI Based on Source Code)

문 중 희 † 이 남 용 ††
(Joong Hee Moon) (Nam Yong Lee)

요 약 GUI 테스트의 자동화는 크게 두 가지로 분류하여 생각해 볼 수 있다. 즉, 회귀 테스트(Regression Test) 자동화와 테스트 자동화(Automated Test)이다. 전자는 테스트 케이스를 수동으로 생성하고 수행만을 자동화하는 의미를 가지는 반면, 후자는 테스트 케이스 생성 자체의 자동화도 포함한다. 점차 테스트에 소요되는 비용이 커지는 상황에서 테스트 자동화 방안을 계속적으로 모색하고 있으나 실제 적용되는 방법은 대부분 회귀 테스트에 한정되어 있다. 즉, 테스트를 처음 수행하는 단계에서는 직접 테스트 케이스를 생성하는 작업이 요구된다. 관련하여 기존의 많은 연구들이 상태 전이도를 기반으로 테스트 케이스를 자동으로 생성하는 방안을 제안하고 있으나 이 방법 역시 사람이 상태 전이도를 작성해야하는 문제를 남기게 된다. 본 논문에서는 자동화 범위를 보다 확대하여 소스코드를 기반으로 테스트 케이스를 자동으로 생성하고 수행하는 방안을 소개한다. 논문에서는 디지털 텔레비전에 탑재되는 셋탑 박스 기반의 어플리케이션 프로그램을 대상으로 연구하였으며 기존에 필요했던 수작업이 없이도 테스트 자동화를 진행하는 것이 가능하다는 것을 실제 적용 사례로 제시하였다. 물론 본 연구결과를 아직 일반화하여 적용할 수는 없을 것이다. 그러나 기존의 테스트 자동화 기법 및 연구들과 비교하여 본 연구결과는 수작업의 양을 보다 줄일 수 있었고 이후 완전한 테스트 자동화 또한 가능하다는 것을 보였다는데 그 의의가 있을 것이다.

키워드 : GUI 테스트 자동화, 자동화 테스트, 회귀 테스트 자동화, 매뉴얼 테스트

Abstract A GUI automated test can be divided into two areas. The first one is a regression test automation and the second one is an automated test. The former includes generating test cases manually and executing them automatically but the latter includes both generating test cases and executing them automatically. Costs of a software test are increasing more and more. Many companies are searching for a test automation method but most used things are limited to regression test automation. So, when testing at first, there should be test cases which are drawn up by a human. This paper explains to make test cases based on a source code and execute them automatically. In this paper, the study proceeds with a digital television set-top box application and explains to test without any effort of human. Of course, this study is far from a realization to industries. But this paper has a contribution at reducing more human efforts than the previous regression test automation and showing that later, fully automated test can be possible.

Key words : GUI test automation, Automated test, Regression test automation, Manual testing

1. 서 론

업계에서 GUI를 테스트 하는 방법은 크게 두 가지로 진행이 되고 있다. 첫 번째는 사람이 리모컨 등을 사용하여 직접 키를 입력하고 결과를 확인하는 방법이다. 두 번째는 전문 테스트 자동화 도구를 사용하는 방안이다. 혹자는 후자의 방법을 테스트 자동화 방안이라고 얘기할 수도 있을 것이다. 그러나 후자의 방법 또한 다시 두 가지로 분류하여 생각해 볼 수 있다. 첫째는 테스트 자동화(Automated Test)이며 둘째는 회귀 테스트 자동화(Regression Test Automation)이다[1]. 전자는 테스트 케이스를 자동으로 생성하고 수행하는 모든 절차의 자

† 정 회 원 : 숭실대학교 소프트웨어공학과
jhmoon77@gmail.com
†† 정 회 원 : 숭실대학교 소프트웨어공학과 교수
nylee@ssu.ac.kr
논문접수 : 2008년 7월 14일
심사완료 : 2009년 7월 17일

정보과학회논문지: 소프트웨어 및 응용 제36권 제9호(2009.9)

동화를 가리키는 반면 후자는 테스트 케이스는 사람이 직접 작성을 하고 반복적인 수행만을 도구에 의존하는 방안이다. 오늘날 테스트에 소요되는 비용이 커지는 상황에서 업계에서는 테스트 자동화 방안을 계속적으로 모색하고 있으나 실제 적용되는 방법들은 대부분 회귀 테스트 자동화에 한정되어 있다. 즉, 테스트 케이스를 누군가가 직접 작성을 해야 하는 것이다. 때문에 처음 테스트 케이스를 작성하는 작업에 적지 않은 시간을 소요할 수 있으며 GUI가 자주 변경될 경우에는 이에 대한 비용이 계속적으로 상승하게 될 것이다. 본 논문에서는 GUI 테스트 자동화(GUI Automated Test)방안을 심층적으로 다룬다. 즉, 테스트를 위한 일련의 모든 작업을 사람의 개입이 없이 자동화 하는 방법을 모색하였다. 연구는 다음과 같은 순서로 진행이 되었다. 2장에서는 업계에서 사용되는 대표적인 상용 도구와 기존의 연구 내역들을 소개한다. 그리고 이러한 연구에서의 주요 성과와 함께 아직은 실용화하기에 어려운 사항들이 무엇인지를 설명한다. 3장에서는 본 논문에서 소개하는 테스트 자동화 방안을 소개한다. 소스 코드를 기반으로 GUI 메뉴 트리를 어떻게 자동으로 생성할 수 있고 생성된 메뉴 트리를 사용하여 테스트 케이스를 어떻게 생성하고 수행할 수 있는지를 설명하였다. 4장에서는 실제 개발 과제를 대상으로 연구 내용을 적용하였으며 그 과정과 결과를 설명하였다. 마지막 5장에서는 본 연구의 결과가 가지는 의미를 분석하고 설명하였다.

2. 기존 테스트 자동화 도구 및 관련 연구

본 절에서는 업계의 대표적인 테스트 자동화 도구를 소개하고 이러한 도구들이 회귀 테스트 자동화에 국한되어 있다는 것을 지적한다. 그리고 이러한 제약을 해결하고자 연구되었던 사례들을 소개하고 본 연구와 비교한다.

2.1 Quick Test Professional[2]

이 도구는 HP 계열사인 Mercury Interactive에서 개발한 대표적인 GUI 기능 테스트 자동화 도구이다. 테스터는 이 도구를 사용하여 GUI 테스트 시나리오를 VB 스크립트(VB Script)로 변환하고 이를 반복적으로 수행할 수 있다. 하지만 GUI 의 아이템 위치 혹은 테스트 시나리오가 바뀔 때에는 새로 스크립트를 작성해야 하는 번거로움이 있다. 즉, 테스트를 반복적으로 수행하는 행위는 도구에 의존할 수 있지만 초기 테스트 시나리오를 스크립트로 변환하는 작업은 사람의 수작업에 의존해야만 한다.

2.2 Hierarchical GUI Test Case Generation Using Automated Planning[3]

논문에서는 Hierarchical Planning Operator를 기반으로 테스트 케이스를 자동으로 생성하여 수행하는 방안을 제시한다. 이 방법을 사용하면 오퍼레이터(Operator)의 단위를 기존의 방식보다 더 많은 이벤트들의 조합으로 가져가기 때문에 그 수는 적어지고 테스트 케이스를 구성하기 위한 노력이 줄어들 수 있다고 설명한다. 또한 초기 상태(Initial State)와 목적 상태(Goal State) 간의 경로를 기존과는 다르게 다양하게 가져갈 수 있다는 것을 주요 성과로 제시한다. 그러나 이 방법 또한 테스트 자동화 도구가 해 주는 작업과 사람이 직접 해야 하는 작업을 필요로 한다. 즉, 사람에 의해서 Hierarchical Planning Operator를 정의하는 일과 초기 상태(Initial State)와 목적 상태(Goal State)를 설정하는 일이 있게 된다. 때문에 이러한 사전 작업에 따른 비용이 작다면 테스트 자동화의 효과는 커지겠으나 그렇지 못하다면 테스트 자동화의 효과는 작아질 것이다. 본 논문에서는 테스트 케이스 자동 생성 이전의 사전 작업을 보다 최소화하는 방안을 제시하고자 했다는데 차별화된 의의가 있을 것이다.

2.3 A Method to Automate User Interface Testing Using Variable Finite State Machines[4]

논문에서는 VFSM(Variable Finite State Machine)을 소개하고 이를 통해서 기존의 FSM(Finite State Machine)과 비교하여 상태 다이어그램(State Diagram)의 복잡도를 크게 낮출 수 있다고 설명한다. 그리고 VFSM에서 FSM으로의 자동 변환이 가능하며 이를 실제 시스템의 사용자 환경(User Interface)에 적용하여 효과를 보인다. 이 논문은 2.1장의 도구가 가지는 회귀 테스트(Regression Test)와는 다르게 테스트 케이스를 자동으로 생성하는 방안이라는 면에서는 긍정적인 효과를 나타낼 수 있었다. 그러나 VFSM을 어떻게 구성할 것인가에 대해서는 차후 연구 과제로 남기게 되어 여전히 사람의 작업이 필요한 것을 확인할 수 있었다. 즉, VFSM의 작성에 따른 비용이 증가할수록 테스트 자동화의 효과는 감소할 수 밖에 없을 것이다. 본 연구가 상태 전이도를 기반으로 테스트 케이스를 자동 생성한다는 것에서는 일부 유사하다고 할 수 있겠으나 상태 전이도 작성에 대한 역할을 사람이 아닌 자동화의 범주로 가져갔다는 것에 있어서는 차이를 가질 것이다.

2.4 Testing Real-Time Embedded Software using UPPAAL-TRON[5]

논문에서는 Timed Automata를 구성하여 테스트 케이스 생성, 수행 및 결과 판정에 대한 자동화 사례를 제시한다. 그리고 Timed Automata를 만들어 내는 로직 및 사례를 자세하게 설명한다. 논문은 Timed Automata를 만들어 내는 알고리즘을 제안하였다는데 큰 의의가 있을 것이다. 사실 기존의 많은 연구들이 테스트

케이스 생성에 앞서 준비되어야 하는 상태 전이도 등을 어떻게 만들지에 대해서는 은연 중에 사람의 작업으로 남기고 있었기 때문이다. 그러나 논문에서 제안한 알고리즘을 기반으로 Timed Automata가 최종적으로 어떻게 산출되었는지를 보여주지 못해서 궁금증을 남기게 되었다. 또한 논문에서는 테스트 대상 소프트웨어(IUT)와의 교신을 위해서는 Physical API를 사용해야한다고 설명을 하고 있다. 이는 기존의 소스 코드에 테스트를 위한 코드를 별도로 삽입하는 작업을 필요로 하는데 매우 민감한 사항이겠으나 이에 대한 구체적인 설명을 하고 있지 않다. 본 논문에서는 테스트 대상 소스 코드에 대한 인스트루멘테이션(Instrumentation)에 대해서 구체적으로 설명하였고 산출된 결과 또한 있는 그대로 제시하였다는데 비교에 대한 의의를 가질 수 있을 것이다.

3. 연구 내용

먼저 GUI 테스트 자동화가 어떠한 작업들을 포함하게 되는지 그 개요를 설명한다. 그리고 자동으로 메뉴 트리를 생성하고 이를 기반으로 테스트 케이스 또한 자동으로 생성할 수 있다는 것을 보인다. 마지막으로 생성된 테스트 케이스를 수행하여 결과를 출력할 수 있다는 것을 설명한다.

3.1 GUI 테스트 자동화 개요

그림 1은 본 논문에서 고안한 테스트 자동화 방안의 개요를 나타낸다. 먼저 소스 코드를 대상으로 GUI 메뉴 트리를 자동으로 생성하는 작업을 수행한다. 생성 작업은 소스 코드를 빌드한 후 생성된 실행 프로그램을 동적으로 수행시키면서 추출되는 로그 메시지를 기반으로 행해질 수 있다. 또한 소스 코드 자체를 정적으로 분석하여 생성할 수도 있다. GUI 메뉴 트리가 생성되면 이를 기반으로 각 메뉴 아이템과 이들 간의 상관 관계를 분석하여 XML기반의 상태전이 다이어그램을 생성한다. 그리고 이를 기반으로 메뉴 트리를 자동으로 돌아다닐 수 있는 테스트 케이스를 생성하게 된다. 테스트 케이스는 임의적으로 혹은 주어진 규칙에 의해서 GUI 메뉴 트리를 돌아다니다가 말단의 메뉴 아이템에 도착했을 때 실제 해당 기능 동작(Action Procedure)을 수행하게 된다. 그리고 이 기능 동작을 수행하는 테스트 코드 또한 자동으로 구현할 수 있을 것이다. 그러나 현재의 연구에서는 이러한 동작은 미리 코드로 준비를 해 놓았으며 향후에는 이러한 동작을 수행하는 테스트 코드 또한 자동으로 생성할 수 있을 것으로 기대한다. 동작을 수행한 결과는 엑셀(Excel) 기반의 테스트 리포트(Test Report)로 남을 수 있게끔 하였다.

3.2 GUI 메뉴 트리 자동 생성 방안

크게 두 가지 방안으로 접근해 볼 수 있었다. 첫째는 대상 소프트웨어를 동적으로 수행한 상태에서 출력되는 로그 메시지를 분석하는 방안이다. 둘째는 소프트웨어를 수행하지 않고 코드 자체를 분석하는 방안이다.

3.2.1 동적 생성 방안

먼저 동적 수행 환경에서 로그 메시지를 추출하기 위해서는 이를 출력하는 코드를 소스코드 내부에 넣어주어야만 한다. 때문에 소스 코드의 향후 변할 여지가 많은 가변부와 변할 가능성이 별로 없는 고정부를 각각 분리하는 작업을 수행하고 고정부 내부에 코드 삽입(Code Instrumentation)작업[6]을 하여 수행 시에 어떤 메뉴 아이템에 포커스가 맞추어져 있는지 알 수 있도록 한다. 고정부에만 코드를 삽입하는 이유는 계속해서 향후 GUI가 바뀌어도 별도의 코드 삽입(Code Instrumentation)작업을 수행하지 않기 위해서이다. 하지만 혹자는 개발 소프트웨어에 따라서 고정부의 범위가 매우 좁을 수도 있을 것이고 더구나 어떤 메뉴 아이템이 선택되었는지를 고정부의 범위만을 조사하여 확인하는 것은 어려울 수도 있을 것이라고 제기할 수도 있을 것이다. 때문에 이럴 경우에는 다음 장의 정적 생성 방안을 생각해 보아야 할 것이다.

그림 2는 메뉴 아이템에 포커스가 맞추어졌을 때 호출되는 콜백 함수 내부에 로그 메시지를 남기는 코드를

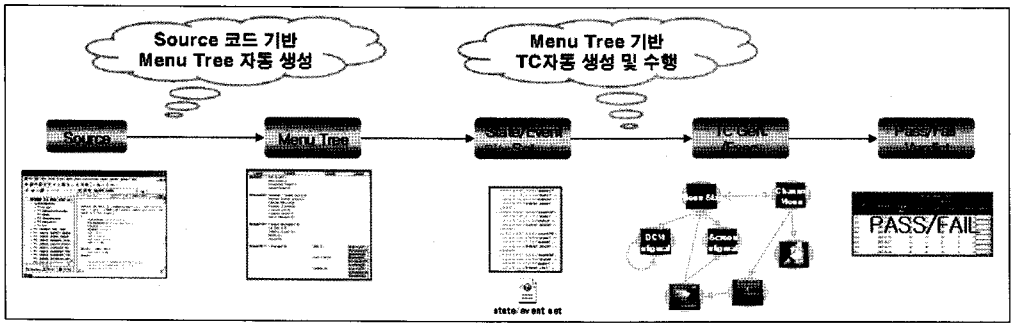

그림 1 GUI 테스트 자동화 개요

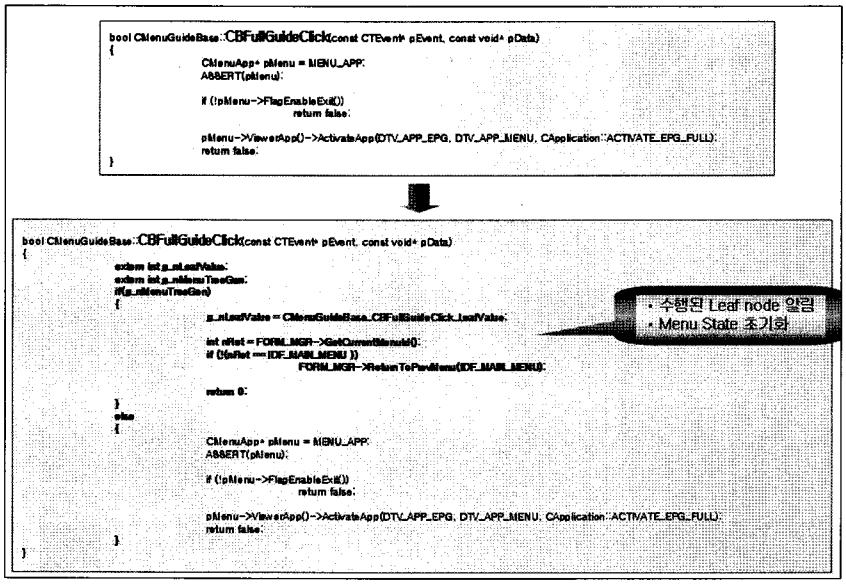

그림 2 Code Instrumentation

삽입한 예를 보인다. GUI 메뉴 트리를 생성하는 테스트 프로그램은 이들 로그 메시지를 확인하여 특정 이벤트 발생에 따라 어떤 메뉴 아이템에 포커스가 맞추어 졌는지를 확인할 수 있다. 그리고 발생한 이벤트와 상관되는 메뉴 아이템간의 관계를 분석하여 메뉴 트리를 생성할 수 있다. 이 경우에 반드시 모든 메뉴 아이템에 대해서 매핑되는 코드를 찾을 필요는 없다. 최하위 말단의 메뉴 아이템에 대응하는 코드만을 확인하여서도 메뉴 트리를 생성하는 작업은 가능하다.

그림 3은 이에 대한 방안을 그림으로 나타낸다. 먼저 초기의 최상단 메뉴 아이템에서 어떤 이벤트를 순차적으로 발생시켰을 때 말단의 메뉴 아이템이 호출되는지

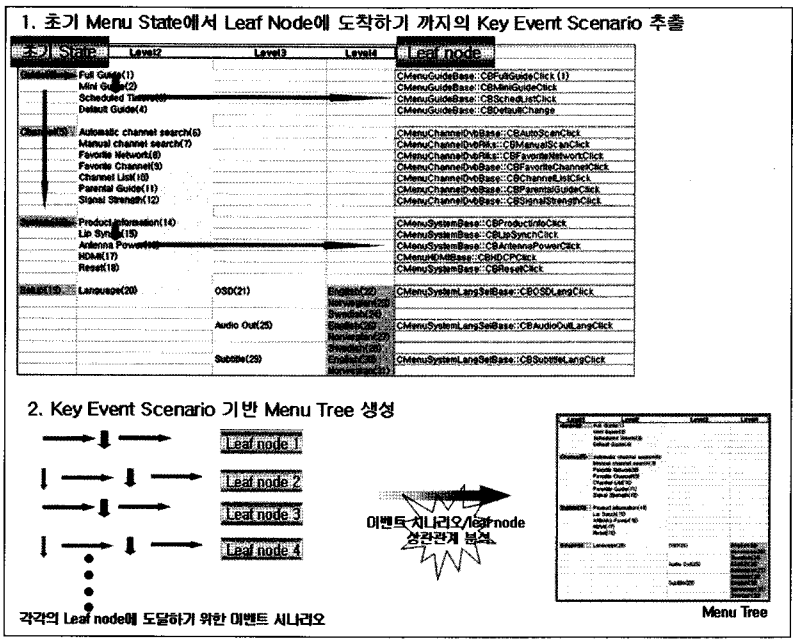

그림 3 동적 수행에 따른 메뉴 트리생성

를 확인한다. 그리고 이벤트 발생 순서에 따른 말단 메뉴 아이템의 호출 관계를 분석하여 메뉴 트리가 어떤 형태인지를 추정할 수 있다. 언급하면 동적 수행 방안은 고정부에 코드를 삽입하여 메뉴 트리를 그려낼 수만 있다면 그 외 다른 조건들 즉, 소프트웨어 아키텍처, 디자인 형태 등을 고려하지 않아도 될 것이다. 그러나 고정부 내의 최소한의 영역에 코드를 삽입하고 이로 인해 출력되는 로그 정보를 기반으로 정확한 메뉴 트리를 생성하는 작업이 어려울 수 있다. 예를 들어 이벤트의 발생에 따라 메뉴 아이템의 이동이 항시 일관되게 변하지 않으면 메뉴 트리를 생성하는 프로그램은 엉뚱한 방향으로 트리를 그려낼 소지가 있다. 또한 트리를 그려내는 프로그램의 로직에 따라 트리를 생성하는 시간이 길어질 수도 있을 것이다. 그리고 소스코드에 코드 삽입을 하는 작업 자체가 다중 쓰레드 환경 등에서는 코드 삽입을 하지 않은 환경과 비교하여 동일하지 않은 결과가 나타날 가능성도 있을 것이다.

3.2.2 정적 생성 방안

정적 생성 방안은 실제 프로그램을 수행시키지 않은 상태에서 소스 코드 자체를 분석하는 것이기 때문에 동적 생성 방안이 가지는 문제점들을 해결할 수 있다. 즉, 이벤트 발생의 효과가 간헐적으로 유실되더라도 메뉴 트리를 생성하는 작업에는 영향을 미치지 않는다. 코드 삽입을 행하지 않기 때문에 기존의 원래 코드를 변형한다는 문제도 제기되지 않을 것이다. 하지만 정적 분석을 하기 위해서는 소스 코드 내부에 GUI 스펙에 대한 정보가 일관된 포멧으로 내장되어 있어야 한다. 메뉴 트리 생성만을 위해서 이러한 구조를 유지한다는 것은 비효율적일 것이다. 하지만 최근 많은 어플리케이션들은 GUI 스펙에 대한 정보를 별도의 인터페이스로 관리하여 GUI의 잦은 변화에 효과적을 대처하고자 노력하고 있다[7,8]. 때문에 이러한 추세를 생각할 때 GUI 스펙 정보를 가지는 인터페이스만을 별도로 분석하여 메뉴 트리를 생성하는 작업은 가능할 것이다.

그림 4는 이에 대한 방안을 그림으로써 나타낸다. 먼저 개발하고자 하는 어플리케이션의 GUI 환경 설정 파일(Configuration File) 혹은 인터페이스를 생성한다. 그리고 이 파일 혹은 인터페이스를 통해서 메뉴 아이템 간의 상관 관계를 분석할 수 있게끔 설계를 한다. 언급하면 이러한 작업을 메뉴 트리 생성을 위해서만 행하는 것은 매우 비효율적인 일이다. 그러나 요즘의 많은 어플리케이션들이 GUI 환경 설정 인터페이스를 별도로 생성하고 관리하고 있기 때문에 이러한 작업은 자연스럽게 진행될 수 있을 것이다. 그림 4에서는 이러한 방안을 적용하였을 때 GUI를 신속하게 변환할 수 있고 GUI의 재사용이 가능하며 또한 GUI 변경에 따라 다른 코드의 수정이 불필요하다는 것을 설명한다. 그리고 마지막으로 메뉴 트리를 자동으로 생성하여 테스트 자동화를 가능하게 할 수 있다는 것을 설명한다.

3.3 테스트 케이스 자동 생성 및 수행

메뉴 트리를 생성하게 되면 메뉴 아이템 및 이들 간의 상관 관계를 정의할 수 있다. 그림 5에서는 메뉴 아이템들 간의 상관 관계를 XML로 나타낸 것을 확인할 수 있다. 그리고 이를 기반으로 테스트 케이스를 생성하고 수행하는 것이 가능하다는 것을 나타낸다.

그림 5는 메뉴 아이템 상관관계 정보를 나타내는 XML 파일을 나타낸다. 각 메뉴 아이템들은 'state'에 숫자가 더해진 이름으로 나타나 있는 것을 확인할 수 있다. 그리고 'STATE KEY'값으로 어떤 이벤트로 관계가 구성되는지를 확인할 수가 있다.

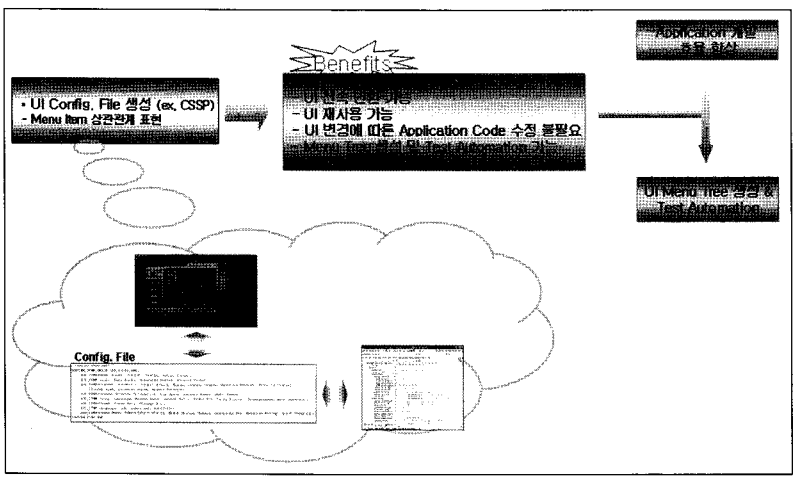

그림 4 정적 분석에 따른 메뉴 트리 생성

```
- <EVENT EVENT_NAME="event5">
    <STATE CUR_STATE="state0" />
    <STATE TO_STATE="state2" />
    <STATE KEY="EVENT_RIGHT" />
  </EVENT>
- <EVENT EVENT_NAME="event6">
    <STATE CUR_STATE="state2" />
    <STATE TO_STATE="state1" />
    <STATE KEY="EVENT_UP" />
  </EVENT>
- <EVENT EVENT_NAME="event7">
    <STATE CUR_STATE="state1" />
    <STATE TO_STATE="state2" />
    <STATE KEY="EVENT_DOWN" />
  </EVENT>
- <EVENT EVENT_NAME="event8">
    <STATE CUR_STATE="state3" />
    <STATE TO_STATE="state0" />
    <STATE KEY="EVENT_LEFT" />
  </EVENT>
```

그림 5 XML기반 메뉴 아이템 상관관계

4. 사례 적용

4.1 적용 대상 개발 과제

내장형 OS를 기반으로 탑재되는 셋탑 박스의 어플리케이션을 대상으로 적용하였다.

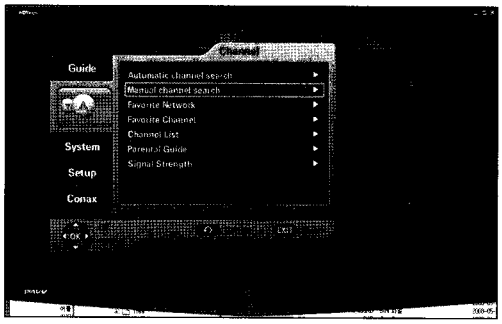

그림 6 셋탑 박스 어플리케이션

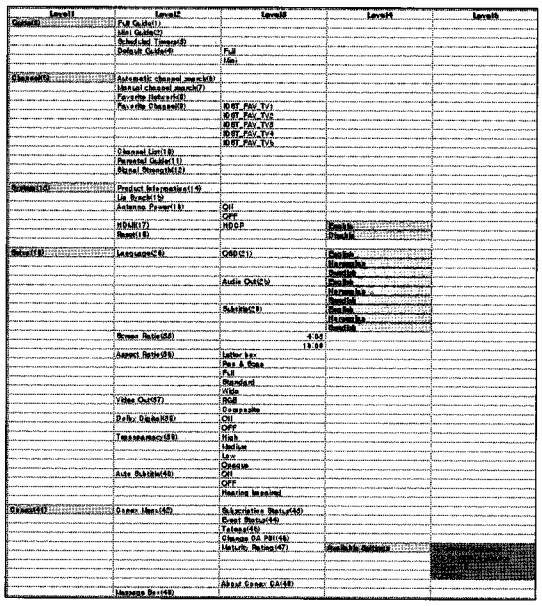

그림 7 메뉴 트리

그림 6은 셋탑 박스 어플리케이션이 보이는 메뉴 화면을 보인다. 그림 7은 어플리케이션의 메뉴 트리를 나타낸다.

4.2 적용 결과

4.2.1 동적 수행 분석에 따른 메뉴 트리 생성

그림 8은 메뉴 트리상에서 말단에 위치한 메뉴 아이템에서 실제 해당 기능 동작 프로시저로 들어갔을 때 호출되는 콜백 함수의 리스트를 나타낸다. 즉, 이들 함수 내부에 그림 2와 같이 코드 삽입 작업을 하여 출력되는 로그 메시지를 분석하게 된다. 그리고 언급하면 그림 3과 같이 이벤트 발생 순서에 따른 콜백 함수 호출 관계를 분석하여 그림 9와 같은 메뉴 트리를 생성하게 되었다. 그러나 실제 적용에 있어서 예상하지 못한 문제들이 발생할 수 있을 것이다. 본 실험에서는 우선 콜백 함수가 항시 해당 기능 동작 프로시저가 수행되는 동시에 호

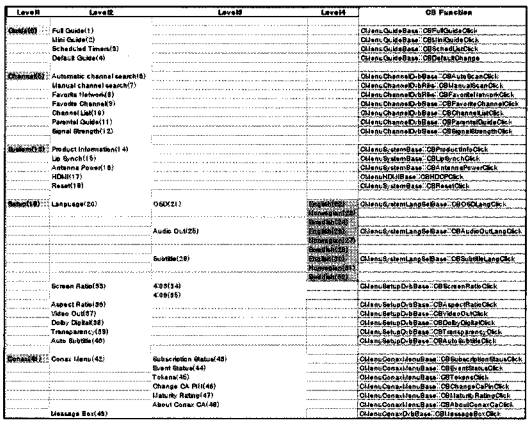

그림 8 메뉴 아이템 Callback Function List

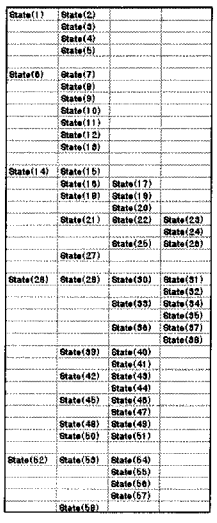

그림 9 동적 수행 기반 메뉴 트리 생성

출되는 것은 아니었다. 즉, 프로시저가 수행되고 특정 상태가 되었을 때에만 콜백 함수가 호출되는 경우가 있었다. 그리고 프로시저가 호출된 이후 의미 없는 이벤트들은 그냥 무시되는 경우도 있었다. 테스트 프로그램은 모든 이벤트가 유효하다고 가정하여 메뉴 트리를 추정하기 때문에 중간에 이벤트가 효력을 상실하여 유실되는 경우에는 문제가 될 수 있었다. 때문에 이러한 경우에 대해서는 대상 소프트웨어의 특성을 고려하여 별도의 부가적인 작업들을 병행할 필요가 있었고 일부에 대해서는 제약사항으로 두어 메뉴 트리의 생성 범위에서 제외할 필요도 있을 것이다. 그림 9는 그림 6의 어플리케이션을 동적으로 수행하고 생성된 메뉴 트리를 나타낸다. 그림 7과 비교하여 많은 부분이 생성되지 않을 것을 확인할 수 있다. 언급하면 이는 대상 소프트웨어의 콜백 함수가 해당 기능 동작 프로시저의 수행과 함께 호출되지 않는 경우가 있었기 때문이다. 만일 초기 개발 단계에서 해당 기능 수행 프로시저에 따른 콜백 함수 호출 관계를 보다 일관성 있게 유지하게 된다면 보다 정확하고 자세한 메뉴 트리를 생성할 수 있을 것이다.

4.2.2 정적 수행 분석에 따른 메뉴 트리 생성

대상 어플리케이션은 그림 10과 같이 GUI 인터페이스를 별도의 환경 설정 파일로 관리하고 있다. 그리고 이를 분석하여 'DEFINE_ITEM_BEGIN', 'ADD_ITEM' 그리고 'SUB_ADD_ITEM'등의 매크로를 사용하여 메뉴 아이템들 간의 상관 관계를 정의한다는 것을 알 수 있었다.

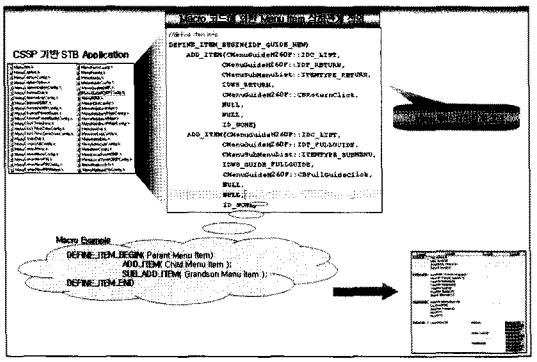

그림 10 정적 분석에 의한 메뉴 트리 생성

그림 11은 생성된 메뉴 트리를 나타낸다. 동적 수행을 하였을 때에는 메뉴 아이템의 이름이 임의로 나타나는 것을 확인하였다. 이는 분석 과정에서 메뉴 아이템의 이름까지는 확인하지 못하였고 이것이 이후 테스트 케이스를 생성하는데 별다른 영향을 못 미치기에 중요시되지 않았다. 그러나 정적 분석에서는 환경 설정 파일에 정의된 메뉴 아이템의 이름이 그대로 생성된 메뉴 트리에 나타난 것을 확인할 수 있다.

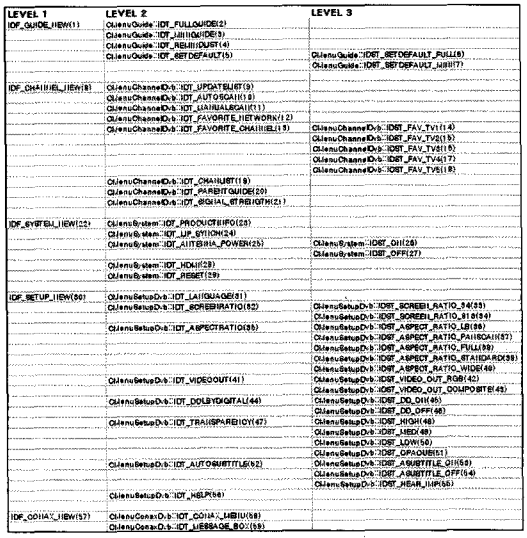

그림 11 정적 분석에 의한 메뉴 트리 생성

그림 12에서 좌측은 정적 분석에 의해 생성된 메뉴 트리를 나타낸다. 그리고 우측은 실제 어플리케이션의 메뉴 트리를 나타낸다. 좌측에서 색깔이 칠해진 부분은 우측과 다르게 나타나거나 생성이 되지 못한 영역을 나타낸다. 이는 환경 설정 파일에서 메뉴 아이템들 간의 상관관계를 분석하지 못하였기 때문이다. 만일 개발 설계단계에서 분석이 용이하게끔 일관된 포멧을 적용하여 환경 설정 파일을 구성하고 메뉴 아이템들 간의 상관관계를 분석할 수 있게끔 한다면 보다 정확하게 메뉴 트리를 생성할 수 있을 것이다.

4.3 테스트 케이스 자동 생성 및 수행

언급하면 먼저 그림 12와 같이 생성된 메뉴 트리를 분석하여 메뉴 아이템들간의 이벤트 상관관계를 분석한다. 그리고 이를 그림 5와 같은 XML파일로 만든다. 즉, XML 파일을 분석하면 테스트 프로그램은 현재 선택된 상태 즉, 메뉴 아이템이 무엇이고 어떤 키를 발생시킬 수 있는지 그리고 키를 발생시키면 어떤 상태로 이동되는지를 알 수가 있다. 때문에 이러한 정보를 기반으로 테스트 프로그램은 메뉴 트리의 아이템들을 계속해서 선택하며 돌아다닐 수 있게 된다. 그리고 말단의 메뉴 아이템에 도착하게 되면 준비된 테스트 프로시저를 수행하게 된다.

그림 13은 그림 6과 같은 메뉴 트리 상의 메뉴 아이템들을 돌아다니다 테스트 프로시저를 수행하는 화면을 나타낸다. 이 테스트 프로시저는 테스트 수행 이전에 미리 준비를 해 두어야 한다. 그러나 향후에는 이러한 프로시저 또한 자동화하는 방안을 모색할 수 있을 것이다.

그림 14는 메뉴 트리를 분석하여 자동으로 테스트 케이스를 생성하고 수행하는 일련의 과정을 그림으로 보

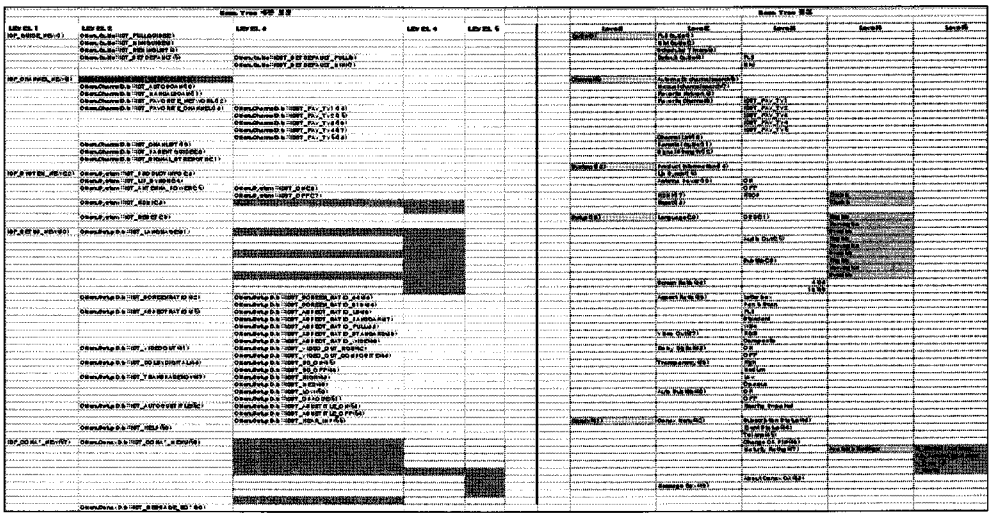

그림 12 생성된 메뉴 트리 비교

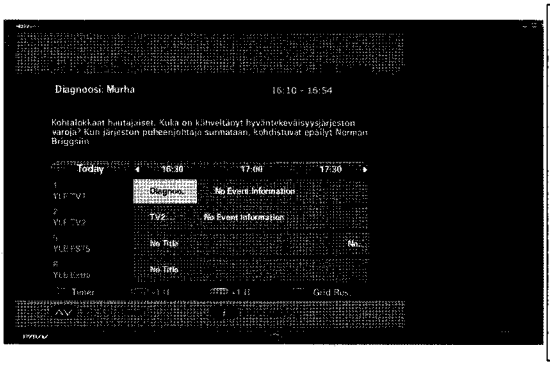

그림 13 테스트 프로시저 수행

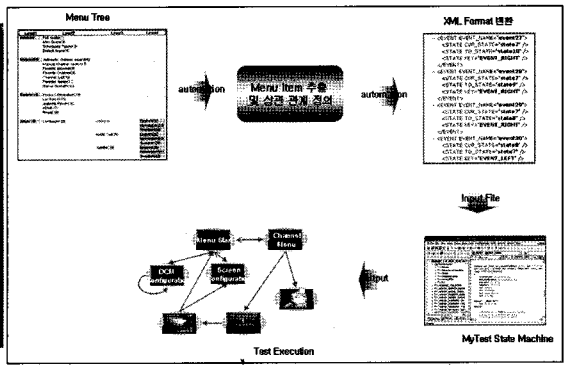

그림 14 메뉴트리 기반 테스트 케이스 자동 생성 및 수행

여준다. 혹자는 테스트 케이스의 단위가 무엇인지 궁금해할 수도 있을 것이다. 테스트 케이스 자동 생성은 메뉴 트리 기반의 XML 상태 다이어그램을 기반으로 계속적으로 메뉴 아이템들을 선택하며 돌아다니는 활동으로 나타난다.

5. 결 론

본 논문에서는 GUI 어플리케이션에 대해서 소스 코드를 기반으로 일련의 모든 테스트 작업들을 자동화 하는 방안을 설명하였다. 기존의 상용 도구 및 대다수 연구들은 회귀 테스트 자동화에 국한되거나 혹은 테스트 케이스 생성을 위한 별도의 수작업을 필요로 하였다. 즉, 전자의 경우에는 사람이 직접 초기 테스트 케이스를 만들어야 하였고 후자의 경우에는 역시 사람이 상태 전이도 등을 작성해야 하는 작업이 필요했다. 그리고 이러한 작업의 시간적 비용만큼 테스트 자동화의 효과는 감소할 수 밖에 없을 것이다. 이와 비교하여 본 연구는 기존의 이러한 작업들을 사람의 개입 없이 자동화 하였다는데 큰 의미가 있을 것이다. 그리고 이러한 자동화 방안을 프로그램의 동적 수행 그리고 정적 코드 기반으로 설명할 수 있었다. 물론 본 연구에서도 사람이 직접 메뉴 아이템 실행 프로시저(Action Procedure)에 대한 테스트 코드를 구현하는 작업이 있었다. 이는 본 논문이 지적한 기존의 제약 사항과는 다른 문제이지만 이에 대한 자동화 영역 역시 앞으로 해결해야 할 과제로 남을 것이다. 또한 기존의 연구들이 자동화 영역을 전반적인 GUI 환경으로 확대한 것에 비해서 본 연구는 GUI 메뉴 트리라는 국한된 사례만을 보인 한계도 있었다. 이 역시 차후 연구에서 점차적으로 풀어야 할 과제로 남을 것이다. 그러나 언급하면 본 연구는 기존의 회귀 테스트 방식에 의한 자동화 방안의 한계 및 문제점을 지적하고 이를 해결하기 위해 발표된 많은 논문들의 방법들 또한 특정 부분에 있어 사람의 수작업을 필요로 한다는 것을 지적할 수 있었다. 그리고 본 논문은 이러한 기존의 논

문들의 제약사항인 사람의 수작업이 필요한 문제에 대해서 방안을 제시할 수 있었다는데 큰 의의가 있을 것이다. 차후 기존 연구들 그리고 본 논문과 더불어 소프트웨어의 테스트 자동화 방안에 대한 연구가 계속해서 지속된다면 점차적으로 보다 사람의 노력을 최소화할 수 있는 이상적인 테스트 자동화 방안을 모색할 수 있을 것이다.

참 고 문 헌

[1] Yury Makedonov, "Manager's Guide to GUI Test Automation," *Software Test & Performance Conference*, November 3, 2005.

[2] Wikipedia Dictionary Available at URL: http://en.wikipedia.org/wiki/HP_QuickTest_Professional, June, 2008.

[3] Memon,A.M., Pollack, M.E., Soffa,M.L., "Hierarchical GUI Test Case Generation Using Automated Planning," *IEEE Transactions on Software Engineering*, vol.27, no.2, Feb 2001.

[4] Richard K. Shehady and Daniel P. Siewiorek, Department of Electrical and Computer Engineering, Carnegie Mellon University, "A Method to Automate User Interface Testing Using Variable Finite State Machines," *27th International Symposium on Fault Tolerant Computing(FTCS '97)*, 1997.

[5] Kim G. Larsen, Marius Mikucionics, Brian Nielsen, Arne Skou, "Testing Real-Time Embedded Software using UPPAAL-TRON," *EMSOFT'05*, September 19-22, 2005.

[6] Wikipedia Directory Available at URL: http://en.wikipedia.org/wiki/Instrumentation_%28computer_programming%29, June, 2008.

[7] Nicola Aloia, Cesare Concordia and Mariateresa Paratore, "Automatic GUI Generation For Web Based Information Systems," Technical Report, *ERCIM Technical Reference Digital Library*, 2003.

[8] Frank Sauer, "A framework for automatic GUI rendering from XML specs," Technical Report, *JavaReport Site*, 2001.

문 중 희

2002년 인하대학교 컴퓨터공학(학사). 2006년 숭실대학교 정보과학대학원(석사). 2006년~현재 숭실대학교 소프트웨어공학과(박사과정). 2002년~현재 삼성전자 DMC 연구소 선임연구원. 관심분야는 소프트웨어 시험, 레거시소프트웨어 재공학, 리팩토링

이 남 용

1979년 숭실대학교 전자계산학(학사). 1983년 고려대학교 경영대학원 경영정보학(석사). 1993년 Mississippi State University 경영정보학(박사). 1999년~현재 숭실대학교 컴퓨터학과 교수. 관심분야는 소프트웨어 품질 시험/인증