Dynamic linking library를 이용한 GUI 자동화 시험 구현 방안

김치열*, 조동훈*, 이홍철**, 권민찬*
*엘아이지 넥스원(주)
**내셔널 인스트루먼츠
e-mail:kcy19underhand@lignex1.com, cho.dong.hoon@lignex1.com,
hongchul.lee@ni.com, kwonminchan@lignex1.com

An GUI Automated Test Technique Based on Dynamic Linking Library

Chi-Yeol Kim*, Dong-Hoon Cho*, Hong-Chul Lee**, Min-Chan Kwon*
*LIG Nex1 Company, Limited.
**National Instruments.

요 약

GUI 테스트 자동화에 대한 연구들이 많이 진행되고 있다. GUI 테스트 하는 방법에 따라 테스트 자동화(Automated Test)와 회귀테스트(Regression Test)가 있는데, 우리는 회귀테스트로 GUI 테스트를 자동화 하려 한다. GUI 자동화 테스트를 위해 시험자동화프로그램으로 테스트 케이스를 시험절차에 따라 생성하고 Dynamic linking library(이하 DLL)의 공유변수마다 GUI 컨포넌트를 정의하여 점검프로그램의 GUI 컴포넌트를 컨트롤 한다. 점검프로그램에서는 DLL 공유변수를 감시하여 해당 GUI 컴포넌트가 On 상태가 되면 기능이 실행되게 구현하였다. 기존의 연구들과 비교하여 최소한의 소스코드 변경과 특정한 툴에 치우치지 않고 통합되어 작동되는 것이 장점이다.

1. 서론

과거의 제품 테스트 방법은 개발자의 의도에 의해 만들어진 소프트웨어와 하드웨어를 품질 관리자가 시험 절차서에 따라 손으로 하나씩 동작시키며 제품 테스트를 수행 하였다. 하지만 이러한 방법은 복잡도가 높은 제품에서는 시험의 방법과 절차가 복잡해지기 때문에 시간과 비용이 증가하게 된다. 이를 해결 해주기 위한 방법은 최근에 많이 출시되고 있는 소프트웨어 테스팅 자동화 전문 툴이다. 테스팅 자동화는 크게 회귀 테스트(Regression Test)와 테스트 자동화(Automated Test)로 분류 할 수 있다. 회귀 테스트는 사람이 직접 테스트 케이스를 작성하여 테스트 순서를 정하여 도구에 정의하여 테스팅을 수행하는 것이고, 테스트 자동화는 테스트 케이스를 자동으로 생성하여 테스트하기 위해서는 소요되는 시간과 노력이 많이 필요하게 된다. 또한 개발자 입장에서는 개발에 몰입하지 자동화 테스트에 대해 강한 의지가 없는 것이 사실이다. 따라서 최근 업계에서는 대부분 회귀 테스트 자동화에 한정을 두게 된다. 이러한 회귀 테스트 방법 역시 쉽지만은 않다. 테스트 케이스는 자동화를 담당하는 관계자가 직접 작성을 해야 하고, 작업순서나 코드가 변경이 되면 테스트 자동화 역시 변경이 되어야 한다. 하지만, 이러한 일련에 작업들은 개발할 때 고려되는 것이지만 개발 일정에 쫓기는 개발자 입장에서는 고려되지 않는다. 따라서 자동화가 되어 있지 않은 제품 테스트는 품질 관리자에게 너무 많은 부담을 지우게 된다. 다시 말하면, 각각 다른 개발자들이 자신이 선호하는 프로그램 툴을 사용하여 제품을 개발하게 되면 품질 관리자는 제 각각 다른 프로그램을 통해 제품의 성능과 각기 다른 기능을 점검하게 된다. 본 논문에서는 테스트 자동화가 고려되어 있지 않은 시스템에서 GUI를 중심으로 한 테스트 자동화 방안을 제안하려 한다. 2장에서는 시장에 소개되어 있는 자동화 도구에 대한 설명과 한계 점을 설명할 것이다. 3장에서는 이미 구현되어 있는 점검프로그램에 대해 GUI 시험 자동화 방법을 제시하고 끝으로 4장에는 적용 사례를 대상으로 개발 과정과 결과를 설명할 것이다.

2. 관련 연구 및 상용화 자동화 Tool

먼저 현재 시장에 배포되어 있는 대표적인 테스트 자동화 툴의 소개와 기능에 대해 소개할 것이다. 대부분 자동화 툴들은 회귀 테스트를 중심으로 제약되어 있거나 점검프로그램을 실행만 시키는 툴도 있다. 또한, 소스 코드를 일부 수정하여 테스트 케이스를 생성하는 연구도 있다. 이러한 기능을 가지고 있는 자동화 툴과 연구 사례를 소개하여 본 논문에서 연구한 방법을 비교 한다.

2.1 National Instrument TestStand

National Instrument(NI)에서 추천하는 자동화 툴은 TestStand와 LabView 조합이다. NI TestStand는 테스트 시퀀스를 제공해주며, LabVIEW는 TestStand의 테스트 스퀀스에 해당하는 시험을 수행하게 된다. NI 제품의 장

점은 소프트웨어와 하드웨어 간에 통합을 쉽게 만들어준다. 개발자가 계측이 필요한 하드웨어에 대해 고민 없이 소프트웨어 계층에서 관찰 가능하게 만들어주는 것이 장점이다. 즉, 테스트에 필요한 광범위한 데이터 수집이나 계측 등을 통해 시험 데이터를 얻게 된다. 제품을 테스트하기 위한 점검용 프로그램이 LabView로 만들어진 프로그램으로 구성이 되어 있을 경우 NI에서 제공해주는 패키징 프로젝트 라이브러리를 통해 구현이 쉬워진다. 하지만, 앞에서도 언급했듯이 기존에 이미 구현되어 있는 점검프로그램을 가지고 자동화하기엔 한계점이 있다. 예를 들어 MFC로 만들어진 점검 프로그램의 GUI를 제어할 수는 없다. NI 제품을 제외한 타사 제품군을 적용시키기 위해서는 실행파일을 Link 시켜 실행만 가능하게 만들어져 있다.

2.2 소스코드 기반 GUI 테스트 자동화 기법

논문[3]에서는 소스코드(Source Code)를 기반으로 자동으로 메뉴 트리를 생성하고 이 결과를 가지고 테스트 케이스를 생성하는 내용이다. GUI 메뉴 트리를 자동으로 생성하는 방법은 두 가지인데 소스 코드에 출력할 로그 메시지를 함수 내부에 넣어주고, 점검 소프트웨어를 동적으로 수행한 상태에서 출력하는 로그 메시지를 분석하는 방법이다. 이러한 방법에 문제점을 대비해 정적 생성 방법을 제시하였다. 정적 생성 방법은 실제 프로그램을 수행시키지 않은 상태에서 소소 코드 자체를 분석하는 것이다. 소스 코드에 임의의 코드를 삽입하지 않아 기존의 코드를 변형시키지 않는 점도 있지만, 소스 코드 내부에 GUI 스펙에 대한 정보를 알아야 한다. 결론적으로 메뉴 트리를 만들기 위해서는 소스 코드를 기반으로 메뉴 트리를 생성하여 점검자가 원하는 시험을 순차적으로 할 수 있다는 것을 보여 주었다.

3. 제안하는 테스트 자동화 방안

본 연구에서는 국방 산업의 특성상 제품에 대한 테스트는 국방 규격으로 정해져 있고, 다양한 툴로 만들어진 점검 프로그램을 자동화 시험이 가능하게 만드는 것을 목표로 한다. 시험 자동화에 필요한 테스트 케이스의 생성은 NI에서 제공하는 TestStand를 사용하여 GUI 자동화를 구현하였다. TestStand에서 문제점은 같은 NI사 제품군에 대해서는 GUI 제어는 가능하지만, Window 계열의 대표적인 툴인 MFC의 GUI 컴포넌트 제어에 대해서는 지원이 되지 않았다. 본 논문에서는 이러한 문제점을 해결하고자 DLL을 이용하여 점검프로그램의 GUI컴포넌트를 제어하여 GUI 시험 자동화 시스템을 구현하려 한다.

3.1 Window의 dynamic linking library

DLL Windows 어플리케이션이 이용할 수 있는 실행 가능한 기능 또는 데이터의 라이브러리이다. DLL의 좋은 예제는 하드웨어 드라이버이다. 어플리케이션 소프트웨어와 하드웨어 사이의 인터페이스는 DLL을 통해 이루어지기도 한다. 보통 DLL은 프로그램이 DLL에 정적 또는 동적 연결을 생성함으로써 액세스하는 특정 기능을 하나 이상 가지고 있다. 정적 연결은 프로그램 실행 중에도 변하지 않는 채로 있으며 동적 연결은 필요에 따라 프로그램이 생성한다. DLL의 또 다른 기능은 여러 어플리케이션 사이에 데이터를 공유할 수 있게 만들어준다.. 본 논문에서는 제품의 검증을 위해 사용되는 다수 어플리케이션의 컴포넌트를 제어하기 위해 DLL을 사용하였다. Microsoft 사의 대표적인 개발 툴인 MFC에서 공유 DLL을 생성시키는 방법은 MSDN을 참조하기 바란다.

3.2 DLL을 이용한 점검 프로그램 설계

그림1은 제품을 검증하기 위해 사용되는 여러 가지 점검 프로그램을 DLL을 통해 컨트롤 하는 방법을 나타내었다. 시험 자동화 프로그램을 사용하기 이전에는 점검 프로그램 1,2,3을 차례로 점검자가 프로그램 GUI에 있는 입력값과 버튼을 점검 절차에 따라 키보드로 입력하고 마우스로 움직이면서 점검을 하였다. 하지만, 시험 자동화 프로그램을 사용하면 점검에 사용되는 입력 값을 한번만 입력해 놓으면 시험 시작과 함께 자동으로 시험을 진행하게 된다. 예를 들어 설명하면, 점검자가 필요한 점검 내용을 시험 목록에 추가 시킨다. 점검자가 점검 시작을 클릭하면, 시험 목록에 나타난 시험 목록대로 진행하게 된다. 진

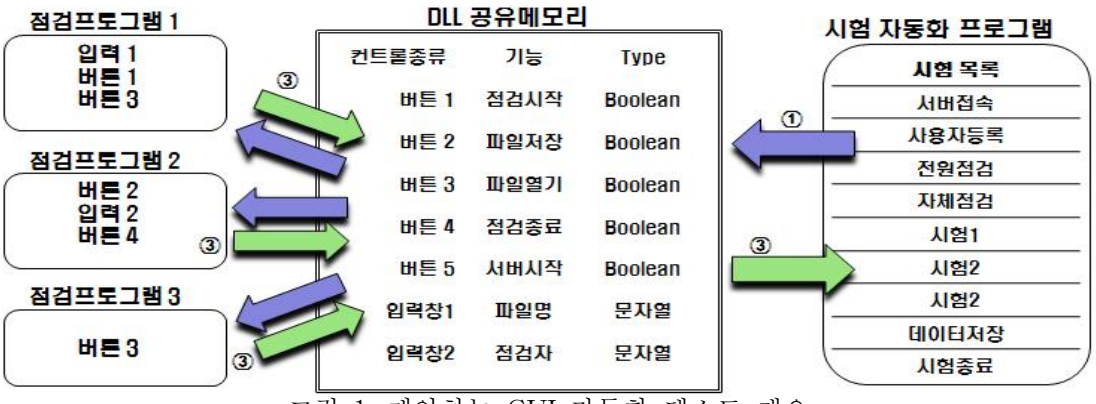

그림 1. 제안하는 GUI 자동화 테스트 개요

행되는 방법은 TestStand의 테스트 케이스가 DLL 공유 메모리에 접근하여 DLL 공유 메모리에 변수 값을 변환시킨다.(그림1의 ①) 각각의 점검 프로그램은 자신의 GUI 컴포넌트에 해당하는 DLL 공유 변수들을 감시하여 DLL 공유 변수가 ON이 되면 해당 GUI 컴포넌트의 함수나 메소드를 호출하게 하면 된다. 함수나 메소드가 종료되면 결과 값을 시험 자동화 프로그램에게 전달하고(그림1의 ③), 다음 테스트 케이스가 진행 된다.

3.3 DLL을 이용한 점검 프로그램 구현 방법

시험 자동화 시스템을 구축하기 위해서는 DLL 공유 메모리와 시험자동화프로그램을 구현해야 하고 점검 프로그램을 일부 수정해야 한다. 먼저, 점검프로그램과 시험자동화프로그램을 연결시켜 줄 DLL 공유 메모리를 정의하고 DLL 공유 변수를 만들면 된다. 점검 프로그램별 GUI컴포넌트 정의 내역은 표 1과 같다.

<표 1> GUI컴포넌트에 따른 공유변수 정의

점검프로그램	컴포넌트종류	데이터형	변수명
프로그램1	Button	Boolean	g_P1btn1
프로그램1	Button	Boolean	g_P1btn3
프로그램1	Edit	Char *	g_P1edt1
프로그램2	Button	Boolean	g_P2btn2
프로그램2	Edit	Char *	g_P2edt2
프로그램2	Button	Boolean	g_P2btn4
프로그램3	Button	Boolean	g_P3btn5

'프로그램1'에서 'Edit' 컨트롤은 사용자가 입력할 입력창이고, 'Button' 컨트롤은 On/Off 타입을 가지게 된다. 표 1에서 정의된 DLL 공유변수를 Microsoft의 개발 툴인 Visual C++ 기준으로 정의하고 만드는 방법은 그림2와 같다. 표 1처럼 정의한 각 변수에 대해 그림 2에서는 코드 영역에서 변수를 선언하였다. 외부 프로그램에서 변수들을 접근하기 위해 구현된 함수들은 함수명 앞에 'Set', 'Get'으로 정의하였다. 입력창은 문자형 배열로 선언하였고, 버튼의 On/Off에 대해서는 정수형으로 선언하여 공유변수에 값을 변경하고, 읽어가는 기능을 구현하였다. 시험 자동화를 위해 점검프로그램의 일부분을 수정해야 한다. 앞에서 언급했던 소스코드 기반의 GUI 테스트는 실제로 운용되는 함수에 로그 정보를 출력할 코드를 삽입하여 GUI가 변경되었을 때, 코드 수정이 불가피 하다. 본 논문에서는 GUI 테스트에 사용될 함수를 고정시켜 이러한 문제점이 없도록 한다. 또한 점검프로그램에 GUI를 컨트롤하기 위해 별도의 쓰레드(Thread)를 만들어 시험 자동화 프로그램이 전달하는 정보를 인식하여 작동하려 한다. GUI를 컨트롤하기 위해 우리는 점검프로그램에 쓰레드를 만들었다. 쓰레드는 각 점검 프로그램에 존재하고, 표1에서 정의된 공유변수들이 On 상태가 되었는지 여부를 판단하게 된다. 시험자동화프로그램의 테스트 케이스에서 DLL 공유 변

```
#pragma data_seg(".share")
int         g_P1btn1;
int         g_P1btn3;
char        g_P1edt1[100];
int         g_P2btn2;
int         g_P2edt2;
int         g_P2btn4;
char        g_P3btn5[100];
#pragma data_seg()
#pragma comment(linker,"/SECTION:.share,RWS")
extern "C" _declspec(dllexport)
            void SetP1btn1Info(int mode, char *p1edt1){
            g_P1btn1 = mode;
            for (int i=0; i < Data_Length; i++){
                        g_P1edt1[i] = p1edt1[i];
            }
}
extern "C" _declspec(dllexport)
            void GetP1btn1Info(int *pmode, char *p1edt1){
            *pmode = g_P1btn1;
            for (int i=0; i < Data_Length; i++){
                        p1edt1[i] = g_P1edt1[i];
            }
}
```

그림 2. DLL 공유 변수 선언 방법

수를 On상태로 변환 시키면 점검프로그램 쓰레드는 이 상태를 확인하여 점검프로그램의 기능이 정의된 함수 또는 메소드를 호출하게 된다. 이렇게 되면 실제로 사용자가 마우스로 해당 버튼을 클릭하는 경우와 동일하다. 주의할 사항은 GUI의 변경을 고려하여 해당 기능에 대한 함수명은 고정되게 정의하면 편리할 것이다. 그림 4는 점검프로그램에 정의된 쓰레드 내용이다.

```
UINT ThreadCtrlUnit(LPVOID pParam) {
    while (TRUE) {
        pGetFlagodeType =
(void (*)(int *))GetProcAddress(hInst,"GetFlagModeType");
        (*pGetFlagodeType) (&nBtn1);
        switch (nBtn1)
        {
            case P1BTN1 :    기능1함수()
            case P1BTN3 :    기능3함수()
            case P2BTN2 :    기능2함수()
            case P2BTN4 :    기능4함수()
            case P3BTN5 :    기능5함수()
        }
    }
}
```

그림 4. 추가된 쓰레드 기능

3.4 시험 자동화 프로그램 구현 방법

우리가 개발한 시험자동화프로그램은 NI의 TestStand4.2 버전이다. TestStand 툴은 앞에서 소개했듯이 LabView로 만들어진 파일의 실행과 다른 툴로 만들어진 파일의 실행이 가능하다. TestStand에서는 "Setup-Main-Cleanup" 세단계로 구성되어 있고, 각 단계별로 품질관리자가 시험절차서에 따라 테스트 케이스를 생성하면 된다. 테스트 케이스의 작성 방법은 LabView를

이용하여 원하는 DLL 공유 변수의 상태 값을 변경하면 된다. 또한 잘 운용된 것인지 확인 메시지까지 받게 되면 다음 테스트 케이스를 진행하면 된다. 그림 5는 시험자동화프로그램의 슈도(Psudo)코드를 나타낸다

```
while (TRUE) {
    switch (nTestCase) {
        case 1 :
                기능1에 대한 공유변수 ON();
        case 2 :
                기능2에 대한 공유변수 ON();
        case 3 :
                기능3에 대한 공유변수 ON();
        case 4 :
                기능4에 대한 공유변수 ON();
        case 5 :
                기능5에 대한 공유변수 ON();
        case 6 :
                시험성적서생성();
    }
    각 기능에 대한 응답 대기();
    Result = 시험결과();
    if (Result)
        결과데이터저장 ();
        nTestCase = 다음TestCase결정함수();
    else
        계속진행여부 ();
}
```

그림 5. 시험자동화프로그램 슈도코드

그림 6은 우리가 개발한 시험자동화프로그램의 메인 프로그램이다. '시험선택항목'은 자동화 시험의 순서가 순차적으로 내열 되어 있다. 테스트 케이스가 '추가/삭제'가 가능하다. 간단한 테스트를 위해 전체를 테스트 할 필요는 없다. 그리고 테스트 케이스 순서도 쉽게 정해주기 위해 테스트 케이스를 아래/위로 이동이 가능하다. 그리고 제품 디버깅에 필요한 RS-232 시리얼 통신이 필요하면 메인 프로그램에서 'Serial' 버튼을 클릭하면 된다. 그림 7에서 시리얼 통신 프로그램은 품질관리자 편의를 위해 명령어를 버튼으로 생성하여 마우스로 클릭을 하면서 운용하게 만들었다. 메인 프로그램에서 테스트 케이스 목록이 결정이 되면 'Start' 버튼을 클릭하여 시험을 시작한다. 테스트는 순차적으로 실행이 되고 테스트 케이스에 대해 결과 값을 기다린다. 시험자동화프로그램은 시험 결과에 에러가 없다면 결과 값을 시험성적서를 생성하기 위해 저장한다. 만약 시험 결과에 에러가 발생하면 다시 수행할지 아니면 종료할지 시험자가 결정하게 한다.

4. 결론

본 논문에서는 DLL을 사용하여 점검프로그램의 GUI 시험 자동화를 구현하였다. 이전 연구에서는 상용도구를 사용하거나 회귀 테스트를 구현하여 제한된 범위에서 GUI 테스트 자동화를 구현하였다. 하지만, GUI의 변경에 따른 소스코드 변환 작업이 이루어지는 단점이 있다. 우리는 이러한 단점을 보완하여 최소한의 점검프로그램의 수정으로 GUI 테스트가 가능하게 구현하였고, 테스트 케이스는 상용화 도구인 TestStand를 이용하여 구현하였지만, 점검프로그램이 만들어진 툴의 종류에 구애 받지 않고 점검프로그램이 제어 가능하다는 것을 보여 주었다. 차후 연구에서는 테스트 케이스를 생성하는 방법에 있어서 상용화 도구를 사용하지 않고 만드는 방법을 연구할 것이며, 시험 자동화를 고려한 점검프로그램 개발 방법에 대해서도 연구할 계획이다.

그림 6. 시험자동화 메인 프로그램

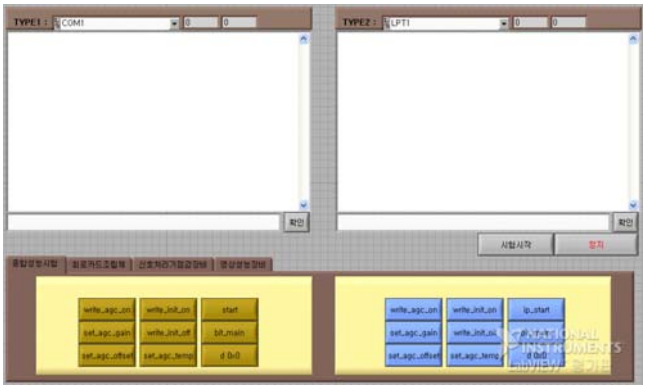

그림 7. 사용자 편의를 위한 시리얼 통신 프로그램

참고문헌

[1] 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.

[2] 이정규, 국승학, 김현수, "시나리오의 자동 생성을 통한 GUI 테스트 케이스 생성 방법", 한국정보과학회 정보과학회논문지 : 소프트웨어 및 응용 제 36권 제1호, pp.45-53, 2009. 1.

[3] 문중희, 이남용, "소스코드기반의 GUI 테스트 자동화 기법의 구현", 한국정보과학회 정보과학회논문지 : 소프트웨어 및 응용, 제36권 제9호, pp.697-705, 2009. 9.

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

[5] NI TestStand, http://www.ni.com/teststand/ko/