韓國 컴퓨터情報學會 論文誌
第10卷 第3號, 2005. 7.

2005-10-3-1-9

XML기반 테스트 정보를 공유하는 소프트웨어 테스팅 자동화 프레임워크의 설계

정 창 신*, 이 계 임**, 김 종 희***, 정 순 기****

Design of Software Testing Automation Framework sharing Test Information based on XML

Chang-shin Chung*, Kye-Im Lee**, Jong-Hee Kim***, Soon-Key Jung****

요 약

소프트웨어 테스팅 도구를 이용하여 테스팅 프로세스의 전체 또는 일부를 자동처리 함으로써 테스팅 시간의 단축과 비용을 줄일 수 있다. 현재 상용화된 테스팅 자동화 도구들은 상호 호환성을 고려하지 않고 개발되었기 때문에 특히, 테스트 설계 단계에서 생성되는 테스트 케이스(테스트 프로시저, 테스트 데이터 및 테스트 수행 결과의 예측 등)의 정보 공유와 재사용이 불가능하다. 본 논문에서는 테스트 케이스의 정보 공유와 재사용을 위하여 테스팅 자동화 도구들의 통합화 대신에 테스팅 수행과정에서 생성된 테스트 케이스 정보를 다른 테스팅 도구들에서도 접근, 공유할 수 있는 테스팅 자동화 프레임워크를 제안하였다. 제안한 통합 환경 테스팅 자동화 프레임워크의 유효성과 효과성, 그리고 시스템 성능과 테스트 케이스의 재사용성을 입증하기 위하여 기존 3개의 테스팅 도구들을 테스트 시나리오에 따라 실험하였다. 실험한 결과 통합 환경 테스팅 자동화 프레임워크 상에서 테스트 케이스의 재사용을 통해 테스팅 시간의 단축과 비용을 감소시킬 수 있었다.

Abstract

The testing time and cost of developed software can be reduced by automation of the whole or part of testing process. Since the testing automation tools to be used currently have been developed without their interoperability, test case information such as test procedures, test data, and expectation of test results generated at the stage of test execution cannot be shared and reused in other testing automation tools. In order to reduce testing time and cost, in this thesis, we have proposed a software testing automation framework which makes it possible to share and reuse the test case information generated in testing process. To prove the availability and effectiveness of proposed testing automation framework, three testing automation tools that are available in current market were experimented by the test scenario. As a result of experiment, the testing time and cost could be reduced by sharing and reusing the test case information in software testing automation framework.

▸ Keyword : Software Testing, Software Testing Automation Framework, Test Information

---

• 제1저자 : 정창신
• 접수일 : 2005.05.16, 심사완료일 : 2005.06.30
* TTA SW시험인증센터 실장, ** 인천기능대학교 컴퓨터정보과 교수,
*** 선문대학교 교수, **** 충북대학교 전기전자컴퓨터공학부 교수(교신저자)
※ 이 논문은 2005년도 충북대학교 학술연구지원사업의 연구비 지원에 의하여 연구되었음.

Ⅰ. 서 론

소프트웨어의 개발에 소요되는 총 비용의 50% 이상과 총 기간의 50% 정도가 개발된 소프트웨어의 테스팅 작업에 할당되고 있다[1]. 소프트웨어 테스팅의 목적은 소프트웨어의 구성요소들이 잘 조합되어 유효하게 작동되며, 소프트웨어가 요구된 성능을 충족시키고 있는 지를 점검하는데 두고 있다. 또한 테스팅은 소프트웨어가 정해진 요구사항을 만족시키고 있는지, 예상했던 결과와 실제 출력 결과 간에는 어떤 차이를 나타내고 있는 지를 수동 또는 자동 테스팅 방법을 이용하여 소프트웨어를 확인(validation), 평가하는 일련의 작업과정을 의미한다[2]. 따라서 소프트웨어 테스팅 단계에서는 개발된 소프트웨어로부터 결함(fault)내지는 오류(error)를 발견하기 위하여 요구사항 명세서, 설계 및 코딩 결과 등에 대한 많은 검토(review)를 수행해야 한다.

소프트웨어 자동 테스팅 도구는 노동 집약적이고도, 단순 반복적인 테스트 업무의 자동 처리에 사용되고 있으며, 이를 통해 테스팅에 소요되는 많은 시간과 비용을 감소시키고 있다. 그러나 현재 상용화된 자동 테스팅 도구들은 다음과 같은 문제점을 가지고 있다.

첫째, 테스팅 도구들의 통합이 어렵다. 여러 업체에서 개발된 테스팅 도구들을 통합, 이용할 수 있는 실질적인 방법은 거의 없으며, 동일한 업체에서 개발한 도구들 간에도 통합 가능성이 미흡하다.

둘째, 기존 테스팅 도구를 기반으로 새로운 자동 테스팅 도구의 개발이 어렵다. 소프트웨어 테스팅은 애플리케이션 영역별로 기술적인 난이도가 다르기 때문에 상용 도구만으로 모든 종류의 소프트웨어를 효과적으로 테스트할 수 없다. 따라서 기존 테스팅 도구를 수정, 이용하거나, 또는 새로운 도구를 개발하여 이용할 수 있지만 테스팅 도구들이 가지고 있는 기능상의 상호 폐쇄성 때문에 도구들의 변경 및 확장에는 한계가 있다.

마지막으로, 기존 자동 테스팅 도구들은 표준화된 테스트 케이스의 생성을 지원하지 못한다. 테스트 케이스의 생성에 표준 양식을 도입하여 재사용이 가능한 테스트 케이스 정보(테스트 프로시저, 테스트 데이터 및 테스트 수행 결과의 분석)를 생성, 축적 및 이용할 수 있도록 준비하는 것이 소프트웨어 품질 보증 부서의 핵심 업무이지만 기존 테스팅 도구에는 이러한 기능이 부족하다[3][4][5].

위와 같은 문제점을 해결하기 위해서는 다양한 테스팅 자동화 도구들의 통합과 기존 도구들을 기반으로 새로운 테스팅 도구의 개발이 가능해야 한다.

본 논문에서는 테스팅 시간의 단축과 비용을 감소시키기 위하여 테스트 수행과정에서 생성된 테스트 케이스 정보를 다른 테스팅 도구들에서도 공유내지는 재사용할 수 있는 환경을 제공하는 소프트웨어 테스팅 자동화 프레임워크를 제안한다.

본 논문의 구성은 다음과 같다. 제 Ⅱ장에서는 관련 연구로써 상용화된 테스팅 프레임워크의 구조에 대하여 기술한다. 제 Ⅲ장에서는 테스팅 도구들이 공유할 수 있는 테스트 정보의 통합과 운영 방법에 대하여 기술하며, 제 Ⅳ장에서는 본 논문에서 제안한 소프트웨어 테스팅 자동화 프레임워크의 구조 및 내부와 외부 연동 구조에 대하여 기술한다. 제 Ⅴ장에서는 제안한 테스팅 자동화 프레임워크의 성능을 측정하고, 그 결과를 분석한다. 마지막 제 Ⅵ장에서는 본 논문의 결론과 향후 연구과제에 대하여 기술한다.

Ⅱ. 관련 연구

미국 SPC(Software Productivity Consortium)에서는 요구사항 명세상의 하자 제거와 테스팅 자동화를 위하여 모델 기반의 자동 테스팅 기법을 사용하는 테스트 자동화 프레임워크에 대한 연구를 진행하고 있으며, 산업계 컨소시엄인 The Open Group에서는 테스트 자동화 프레임워크로 TETware(Test Environ ment Toolkit) 프레임워크를 발표하였다. 그리고 국제 표준화 기구인 OASIS에서는 XML 기반의 테스트 프레임워크로 ebXML(electronic business XML) 프레임워크를 제안하였다[6][7][8].

본 논문에서 제안하는 소프트웨어 테스팅 자동화 프레임워크와 연구 중인 테스트 프레임워크간의 차이점을 분석하

기 위하여 TETware 프레임워크와 ebXML 프레임워크에 대하여 살펴본다.

2.1 TETware 프레임워크

TETware 프레임워크의 구조는 (그림 2.1)과 같이 테스트 케이스의 개발, 수행 및 관리 기능으로 구성되며, 이들을 라이브러리 형태로 제공한다[6]. 실제로 테스팅 프레임워크를 정의, 구현하여 제공하고 있기 때문에 현재 제시된 테스팅 프레임워크 가운데 가장 발전된 형태를 가지고 있다. 또한 다양한 종류의 테스팅 도구들을 수용할 수 있도록 설계되어 있기 때문에 범용성도 뛰어나다. 그러나 현재 TETware 프레임워크에서 제공하는 라이브러리는 주로 테스트 케이스를 개발, 수행 및 관리하는 기능만을 가지고 있기 때문에 그 외의 많은 기능들은 테스트 엔지니어가 직접 개발해야 된다는 단점이 있다. 특히, 테스트 케이스의 개발에는 C/C++, Java 등을 사용해야 한다.

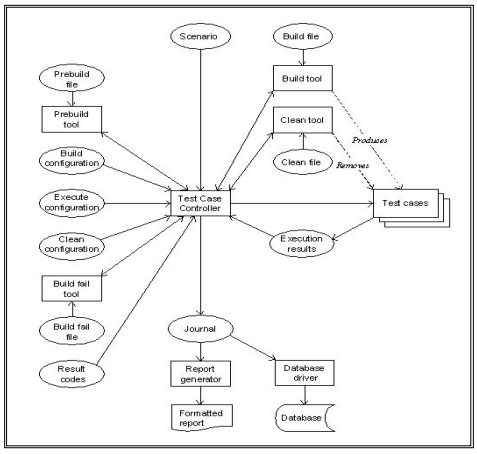

그림 2.1 TETware 프레임워크 구조
Fig 2.1 TETware Framework Structure

2.2 ebXML 프레임워크

국제 표준화 기구에서 상용 테스트 도구들과는 관련 없이 표준 테스트 프레임워크를 제안하였다. 이러한 전자상거래 표준화 기관인 OASIS(Organization for the Advancement of Structured Information Standards)에서는 소프트웨어의 표준 적합성 및 상호 운용성 테스트를 위하여 ebXML 테스트 프레임워크를 수용하고 있으며, 그 구조는 (그림 2.2)와 같다[7].

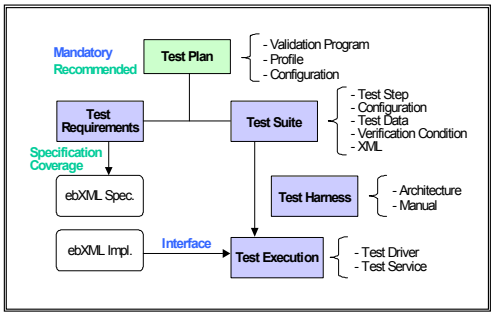

그림 2.2 ebXML 테스트 프레임워크 구조
Fig2.2 ebXML Test Framework Structure

ebXML 테스트 프레임워크는 ebXML 고유의 표준 규격을 기반으로 소프트웨어의 표준 적합성과 상호 운용성을 테스트하는 방법을 정의하며, ebXML 테스팅의 각 유형별로 테스트 동력(harness)을 생성하기 위해 테스트 베드(test bed)와 소프트웨어 컴포넌트들 간의 결합 방법을 정의하고 있다.

테스트 프레임워크의 컴포넌트는 다른 형상(configuration) 또는 테스트 동력과 결합될 수 있도록 설계되었다. 여기서 테스트 동력은 소프트웨어 컴포넌트의 테스팅 시 테스트 실행 결과와 예상 결과의 비교에 사용된다.

ebXML 테스트 프레임워크를 사용하는 소프트웨어 테스팅은 다음과 같다.

- 단계 1 : 테스트 계획 수립(권고사항)
- 단계 2 : 테스트 요구사항 설계(필수사항)
- 단계 3 : 테스트 동력 설계(필수사항)
- 단계 4 : 테스트 슈트 설계(필수사항)
- 단계 5 : 확인 조건 정의(권고사항)
- 단계 6 : 테스트 슈트 실행(필수사항)

본 논문에서는 테스트 프레임워크의 소프트웨어 테스팅 단계 중 단계 4(테스트 슈트 설계 방법)를 적용하여 테스트 프레임워크 상에서 공유하는 테스트 정보를 정의하고 생성한다.

III. 테스트 정보의 통합

본 장에서는 테스트 프레임워크 상에서 테스트 정보를 통합하는 과정 (그림 3.1)을 기술한다.

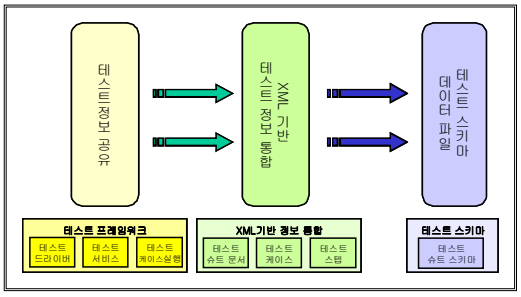

그림 3.1 테스트 정보의 통합 과정
Fig 3.1 Integration Process of Test Information

테스트 정보 통합을 위하여 테스트 프레임워크의 구성요소를 분석하고, 공유가 가능한 테스트 정보와 공유가 불가능한 테스트 정보를 분류한다. 그리고 공유하는 테스트 정보를 XML 문서 기반의 테스트 슈트 문서, 테스트 케이스 및 테스트 스텝의 데이터 구조로 변환시키고, 공유가 가능한 테스트 정보를 기반으로 하는 테스트 케이스 정보의 XML 스키마 구조를 정의하고, 정의한 XML 스키마를 기반으로 하는 테스트 데이터 파일을 생성한다.

3.1 테스트 프레임워크의 구성요소

테스트 프레임워크는 테스트 드라이버, 테스트 서비스 및 테스트 케이스 실행으로 구성된다.

3.1.1 테스트 드라이버

테스트 드라이버는 테스트 케이스의 각 스텝을 구동시키는 역할을 하며, 마크업 언어로 작성된 테스트 케이스를 파싱하고 해석한다.

3.1.2 테스트 서비스

테스트 서비스는 테스트 케이스를 수행하는 액션(action)의 집합이며, 메시지를 조정하는 애플리케이션 계층을 나타낸다.

3.1.3 테스트 케이스 실행

테스트 케이스 실행은 테스트 요구사항의 규격이 제대로 구현되었는지를 확인한다. 여기서 테스트 케이스는 XML 문서를 의미한다. 또한 테스트 케이스는 일련의 테스트 스텝으로 구성되며, 테스트 스텝은 테스트 동력(harness)의 컴포넌트가 수행되는 한개 이상의 오퍼레이션 집합이다. 테스트 스텝은 테스트 케이스 데이터베이스에게 메시지 데이터를 요구한다. 메시지 데이터는 XML 기반 스크립트로써 메시지 선언과 같이 작성된다.

본 논문에서는 테스트 프레임워크 구성요소 중에서 테스트 케이스 실행을 공유와 재사용이 가능한 테스트 케이스 정보로 취급한다.

3.2 공유 정보와 비공유 정보

테스트 프레임워크에서 테스트 정보의 공유 또는 비공유를 구분하기 위하여 테스팅 대상이 되는 소프트웨어 영역, 공유할 수 있는 테스트 정보의 범위 및 공유 정도를 측정하는 테스트 항목/테스트 테크닉 매트릭스를 정의하고 사용한다.

3.2.1 소프트웨어 영역

테스트 정보를 공유할 수 있는 소프트웨어 영역은 테스팅 기술, 특성화된 시스템 및 애플리케이션 테스팅에 따라 분류된다.

본 논문에서는 테스팅 대상이 되는 소프트웨어 영역을 특성화된 시스템 및 애플리케이션 테스팅 영역에 속하는 웹 기반 시스템 테스팅으로 제한한다.

3.2.2 테스트 정보의 범위

소프트웨어 테스팅 자동화를 위해서는 테스트 케이스와 테스트웨어가 준비되어야 하며, 테스트 수행과정에서 사용되는 테스트 정보는 (그림 3.2)와 같다.

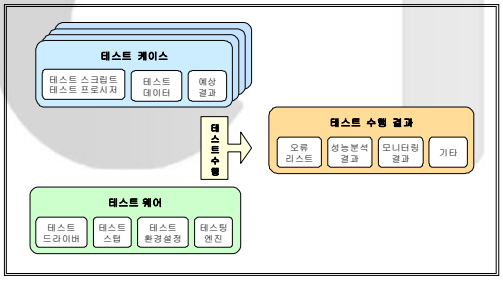

그림 3.2 테스트 수행과정과 테스트 정보
Fig 3.2 Test Execution Process and Test Information

본 논문에서는 테스트 프레임워크에서 공유하는 테스트 정보의 범위를 테스트 케이스 정보로 제한한다.

3.2.3 테스트 항목/테스트 테크닉 매트릭스

테스트 항목에 따라 테스트 기술을 선택하기 위해서는 <표 3.1>과 같은 테스트 항목/테스트 테크닉 매트릭스를 사용하며, 테스트 정보의 공유정도를 측정한다[9].

표 3.1 테스트 항목/테스트 테크닉 매트릭스
Table 3.1 Test Factor/Test Technique Matrix

테스트 테크닉 / 테스트 항목	구조적 시스템 테스팅						기능적 시스템 테스팅							유니트 테스팅
	스트레스	실행	복구	운영	준수성	보안성	요구사항	회귀	에러 핸들링	메뉴얼 지원	시스템 간	제어	병행	
정확성														
파일 무결성														
권한 부여						✓	✓							✓
감사 기록														
트로젠 면속성														
서비스 레벨														
접근 제어						✓								
준수성														
신뢰성														
사용 관리성														
유지 보수성														
이식성														
결합도														
성능														
운영 관리성														

사용자의 접근 제어 및 권한 부여 테스트 항목을 테스팅 할 경우는 구조적 시스템 테스팅 기술을 이용하여 보안성을 체크해야 하며, 기능적 시스템 테스팅 기술을 이용할 경우는 요구사항의 체크와 유니트 테스팅을 수행해야 한다.

3.3 XML 문서 기반의 테스트 정보 통합

테스트 프레임워크에서 공유하는 테스트 정보(테스트 케이스)를 XML 문서 기반의 테스트 슈트 문서, 테스트 케이스 및 테스트 스텝의 데이터 구조로 변환시키고, 각 문서에 따라 테스트 정보의 공유 또는 비공유를 구분한다. 구분한 공유 테스트 정보를 기반으로 하는 테스트 케이스 정보의 XML 스키마 구조를 정의하고, 정의한 XML 스키마를 기반으로 하는 테스트 데이터 파일을 생성한다.

3.3.1 XML 스키마

공유 테스트 정보를 기반으로 하는 테스트 케이스 정보의 XML 스키마 구조 <표 3.2>를 정의한다.

테스트 케이스 정보의 XML 스키마는 테스트 슈트 문서, 테스트 케이스 및 테스트 스텝을 포함하고 있으며, 각 문서의 구성 데이터 중에서 테스트 케이스 정보의 데이터 정의와 관련된 데이터들(TestSuite, TestCase, id, name, TestStep, PutMessage, GET Message)을 스키마 구조 정의에 사용한다.

표 3.2 테스트 케이스 정보의 XML 스키마 구조
Table 3.2 XML Schema Structure of Test Case Information

```
<?xml version="1.0" encoding="UTF-8" ?>
<schema targetNamespace="http://tests" x
mlns:Test="http://tests"
xmlns="http://XMLSchema" version="1.0">

<element name="TestSuite">
<complexType>
<sequence>
  <element ref="Test:TestCase" />
  </sequence>
  </complexType>
  </element>

<element name="TestCase">
<complexType>
<sequence>
  <element ref="Test:TestStep" />
  </sequence>
  <attribute name="id" type="string" />
  <attribute name="name" type="string" />
  </complexType>
  </element>

<element name="TestStep">
<complexType>
<choice>
  <element ref="Test:PutMessage" />
  <element ref="Test:GetMessage" />
  </choice>
  </complexType>
  </element>
  </schema>
```

3.3.2 테스트 데이터 파일

XML 스키마를 기반으로 하는 테스트 데이터 파일을 생성한다. 테스트 데이터 파일은 제안한 테스팅 자동화 프레임워크의 테스트 정보 저장소(TiRE: Test Information REpository)에 XML 문서 형태로 저장하며, 저장한 XML 문서는 XML 문서 집합 단위로 관리한다. <표 3.3>은 XML 문서 기반의 테스트 데이터 파일을 테스트 정보 저장소에 저장한 구조를 나타내며, 관계형 데이터베이스의 구성 요소들과 테스트 데이터 파일의 상호 연관성을 보여준다.

표 3.3 XML 문서 기반의 테스트 데이터 파일의 저장 구조
Table 3.3 Repository Structure in XML Document based Test Date File

관계형 데이터베이스	테스트 정보 저장소(TiRE)
데이터베이스	XML 문서 집합(XML 파일의 집합)
테이블	XML 문서(XML 파일, TiRE 테이블)
레코드	XML 엘리먼트(TiRE 레코드)
필드	XML 애트리뷰트(TiRE 필드)

XML 문서에는 루트 엘리먼트는 제외되고, 동일한 종류의 엘리먼트만이 존재한다. 엘리먼트는 내용을 갖지 않는 빈(empty) 엘리먼트로써 한 개 이상의 애트리뷰트를 갖는다. 실 데이터는 애트리뷰트에 저장한다. <표 3.4>는 제안한 테스팅 자동화 프레임워크의 테스트 정보 저장소에 저장한 테스트 데이터 파일을 보여준다.

표 3.4 테스트 정보 저장소에 저장한 테스트 데이터 파일(XML 문서)
Table 3.4 Stored Test Data File in Test Information Repository(XML Document)

```
<?xml version="1.0" encoding="UTF-8" ?>
<Script_table>
<Script rid=" R786" name=" api3_204" api_no=" 204"
flag=" 1" parameter=" R787" />
<Script . . . />
. . .
</Script_table>
```

Ⅳ. 테스팅 자동화 프레임워크의 설계

본 장에서는 제 Ⅲ장에서 정의한 테스트 케이스 정보를 통합하는 테스팅 자동화 프레임워크의 설계에 필요한 프레임워크의 구조, 외부 및 내부 연동 구조 설계에 대하여 기술한다.

4.1 프레임워크 구조

테스팅 자동화 프레임워크의 구조는 (그림 4.1)과 같이 테스팅 엔진, 테스트 라이브러리, 테스트 정보 저장소 등으로 구성된다.

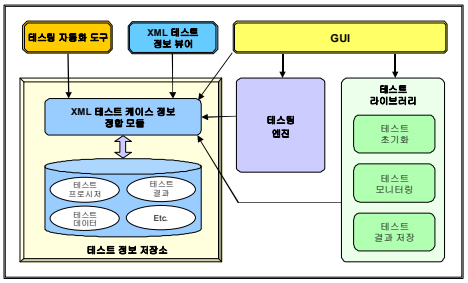

그림 4.1 테스팅 자동화 프레임워크 구조
Fig 4.1 Testing Automation Framework Structure

4.1.1 테스팅 엔진

테스팅 엔진은 테스트 케이스를 실행하고, 실행 결과를 모니터링 하는 역할을 담당한다. 테스트 케이스 정보는 XML로 표현되므로 테스팅 엔진은 XML 문서를 해석하고 실행할 수 있는 기능을 가지고 있어야 한다.

4.1.2 테스트 라이브러리

테스트 라이브러리는 현재 각 테스팅 자동화 도구마다 별도로 제공하고 있다. 따라서 테스팅 자동화 프레임워크는 테스트 케이스의 설계와 개발을 위하여 테스팅 초기화 기능, 테스트 관찰 기능, 테스트 결과의 저장 기능을 갖는 라이브러리를 제공할 수 있어야 한다.

4.1.3 테스트 정보 저장소

테스트 정보 저장소에는 소프트웨어 테스팅 과정에서 생성되는 결과물들이 저장된다. 테스트 결과물의 형식과 의미는 테스팅 자동화 도구에 따라 달라진다. 따라서 테스트 정보 저장소는 기존 테스팅 자동화 도구에 의해 생성되는 테스트 결과물을 수용할 수 있을 뿐만 아니라, 테스팅 자동화 프레임워크 상에서 생성되는 결과물도 수용할 수 있다. 테스트 정보 저장소에 있는 테스트 케이스 정보를 상이한 테스팅 도구들이 접근하고 공유할 수 있도록 하기 위하여 유연성이 뛰어난 테스트 정보 저장소 (그림 4.2)를 설계한다.

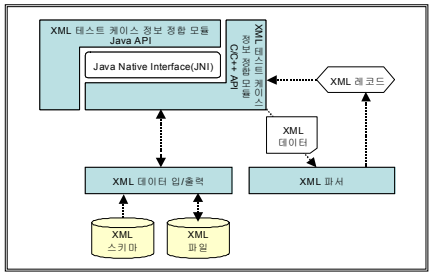

그림 4.2 테스트 정보 저장소 구조
Fig 4.2 Test Information Repository Structure

4.2 외부 연동 구조

테스트 정보 저장소와 외부 모듈과의 연동에는 XML 테스트 케이스 정보 정합 모듈(그림 4.3)을 사용하며, API를 이용하여 데이터를 주고받는다.

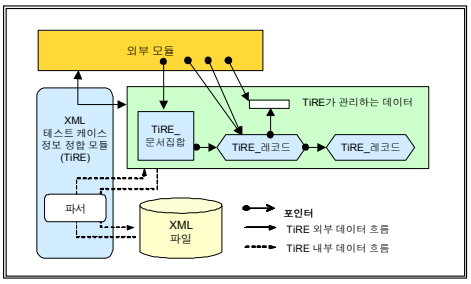

그림 4.3 테스트 정보 저장소의 외부 연동 구조
Fig 4.3 External Interworking Structure of Test Information Repository

4.3 내부 연동 구조

테스트 정보 저장소는 Java API와 C/C++ API 처리 모듈로 구성되며, Java API 처리 시에는 C/C++ API 서브모듈과 연동 (그림 4.4)되어 그 기능을 수행한다.

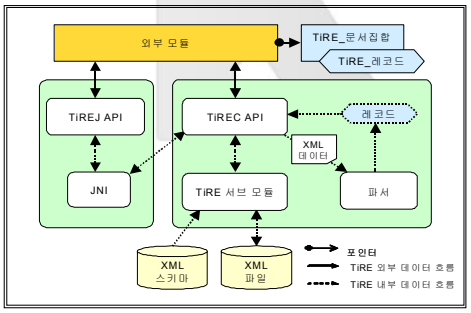

그림 4.4 테스트 정보 저장소의 내부 연동 구조
Fig 4.4 Internal Interworking Structure of Test Information Repository

4.3.1 XML 파일과 XML 파일 정보

TiRE_문서집합(XML 문서집합)은 (그림 4.5)와 같이 각 테이블에 대한 정보를 유지한다. 테이블 정보로는 RID, 파일에서의 레코드 위치 및 크기, 레코드를 가리키는 포인터 (레코드가 메모리에 적재된 경우) 및 free RID 등을 들 수 있으며, 이들은 TiRE_문서집합의 개방(open) 시 적재된다.

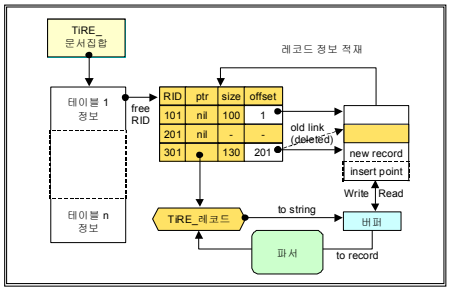

그림 4.5 테스트 정보 저장소의 XML 파일 정보 구조
Fig 4.5 XML File Information Structure of Test Information Repository

4.3.2 테스트 정보 저장소의 내부 데이터 구조

TiRE_문서집합, 테이블 정보, 레코드, 파서 및 버퍼의 데이터 구조는 <표 4.1>과 같다.

표 4.1 테스트 정보 저장소의 내부 데이터 구조
Table 4.1 Internal Data Structure of Test Information Repository

자료구조	필 드	자료형	의 미
struct TiRE_DOC_SET	working_dir	char*	작업 디렉토리
	psr	TiRE_PSR*	XML 파서 정보
	buf	TiRE_BUF*	read/write 버퍼
	table	TiRE_TBL_INFO[]	테이블 정보
struct TiRE_TBL_INF	xf	TiRE_XF*	XML 파일 정보
	scm	TiRE_SCM*	TiRE 스키마 정보
	rit	TiRE_RIT*	레코드 정보 테이블
struct TiRE_RIT	count	int	RID 수
	rid	int[]	레코드 ID
	rec_ptr	TiRE_REC*[]	레코드에 대한 포인터
	size	int[]	블록 크기
	offset	int[]	파일에서 블록 시작위치
	prev	TiRE_RIT*	이전노드
	next	TiRE_RIT*	다음 노드
struct TiRE_PSR	parser	void*	파서에 대한 포인터
	flag	int	플래그
struct TiRE_BUF	size	int	버퍼 크기
	data_size	int	실제 데이터 크기
	buf	char*	버퍼
struct TiRE_XF	path	char*	XML 파일 절대 경로
	fd	FILE*	파일 설명

V. 실험 및 결과 분석

본 장에서는 제 IV장에서 설계한 소프트웨어 테스팅 자동화 프레임워크의 유효성과 효과성을 검증하기 위하여 프레임워크를 3개의 상용 테스팅 자동화 도구에 적용하여 실험을 수행한다. 테스트 시나리오에 의해 실험을 수행하며, 실험 결과를 분석하여 테스팅 자동화 프레임워크의 성능과 테스트 케이스 정보의 재사용성을 평가한다.

5.1 실험 환경

실험에 사용한 시스템은 (그림 5.1)과 같이 구성하였으며, 실험 순서는 다음과 같다.

① I사의 R제품을 사용하여 테스트 스크립트를 생성한다.

② I사의 R제품에서 생성된 테스트 스크립트를 테스트 정보 저장소에 저장한다.

③ 테스트 정보 저장소에 저장된 테스트 스크립트로부터 C사의 A제품과 B사의 C제품(④)에서도 접근할 수 있는 테스트 스크립트를 생성한다.

위와 같은 실험을 수행하여 테스팅 자동화 프레임워크의 유효성과 효과성을 검사한다.

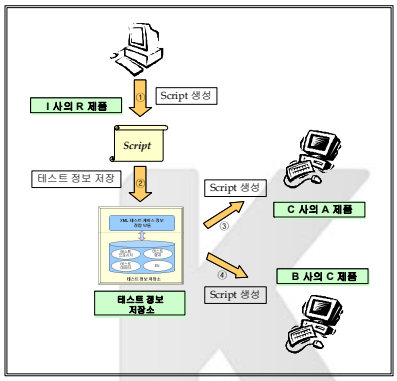

그림 5.1 테스팅용 시스템 구성
Fig 5.1 System Configuration for Testing

본 논문에서는 테스팅 대상이 되는 소프트웨어의 영역을 웹 기반 시스템 환경으로 제한하여 시스템 테스트를 수행한다.

5.2 테스트 시나리오

웹 기반 시스템의 사용자 권한 부여와 접근 제어(인증), 정확성과 파일 무결성 및 성능 테스팅을 위한 테스트 시나리오는 <표 5.1>과 같다.

표 5.1 테스트 시나리오
Table 5.1 Test Scenario

시나리오 번호: 001				
목 적 : 특정 홈페이지 접속 후 사용자 화면 테스팅				
특수 요구 : 자료 구조의 초기화에 필요한 비밀번호 검증, 사용자 아이디와 비밀번호 검증 등				
1. 사용자 아이디와 비밀번호 없이 로그인을 시도한다.				
순서	입력 데이터	예상 출력 데이터	실행결과	비 고
1	홈페이지의 "로그인"아이콘을 누름	사용자 로그인 화면이 보임	-	-
2	사용자 로그인 화면에서 "LOGIN" 버튼을 누름	사용자 로그인 화면이 보임	-	사용자 로그인 화면의 사용자 아이디 필드에 입력 가능한 상태로 전환됨
2. 비밀번호 입력 없이 정상적인 사용자 아이디로만 로그인을 시도한다.				
순서	입력 데이터	예상 출력 데이터	실행결과	비 고
1	사용자 아이디 필드를 누름	사용자 아이디 필드에 캐릿이 깜박거림	-	-
2	정상 사용자 아이디 "cschung" 입력	사용자 아이디 필드 화면이 보임	-	-
3	사용자 로그인 화면에서 "LOGIN" 버튼을 누름	사용자 로그인 화면이 보임	-	사용자 로그인 화면의 사용자 아이디 필드에 입력 가능한 상태로 전환됨
3. 사용자 아이디 입력 없이 정상적인 비밀번호로만 로그인 시도한다.				
순서	입력 데이터	예상 출력 데이터	실행결과	비 고
1	비밀번호 필드 누름	비밀번호 필드에 캐릿이 깜박거림	-	-
2	정상 비밀번호 "cschung" 입력	비밀번호 필드에 문자가 보이지 않음	-	-
3	사용자 로그인 화면에서 "LOGIN" 버튼을 누름	사용자 로그인 화면이 보임	-	사용자 로그인 화면의 사용자 아이디 필드에 입력 가능한 상태로 전환됨
4. 정확성과 파일 무결성 테스트를 시도한다.				
순서	단 계	예상 출력	실행결과	테스트 항목
1	로그인	-	-	정확성
2	R 제품 테스트 스크립트 생성	스크립트 생성	-	정확성/무결성
3	테스트 정보 저장 시스템에 저장	테스트 정보 저장	-	정확성/무결성
4	A 제품 테스트 스크립트 생성	스크립트 생성	-	정확성/무결성
5	C 제품 테스트 스크립트 생성	스크립트 생성	-	정확성/무결성
5. 성능을 평가하기 위한 테스트를 시도한다.				
순서	테스트 등급	CPU 시간	메모리 점유율	비 고
1	Worst-Case	대기 소요시간 부하(%)	%	미승인
2	Unsure	장애 복구시간 부하(%)	%	DB 장애
3	Good/Not bad	응답시간 부하(%)	%	늦은 응답처리
4	Best-Case	처리시간(%)	%	-

5.3 결과 분석

본 논문에서 제안한 소프트웨어 테스팅 자동화 프레임워크의 유효성과 효과성을 입증하기 위하여 세 종류의 테스트 자동화 도구를 테스트 시나리오에 따라 실험하고, 그 결과를 평가하였다. 특히, 테스트 케이스 정보의 공유와 재사용을 중심으로 제안된 테스팅 자동화 프레임워크의 유효성과 효과성을 분석하였다.

5.3.1 권한 부여와 접근 제어 기능의 평가

권한 부여와 접근 제어 기능을 평가하기 위하여 실험용 웹 기반 시스템 상에서 세 종류의 테스팅 도구와 테스팅 자동화 프레임워크에 대하여 실험을 수행하였으며, 실험 결과는 <표 5.2>와 같다.

<표 5.2>에 의하면 테스팅 자동화 프레임워크에서 시스템의 접근 제어는 다른 테스팅 도구들 보다 안전성을 가지는 것으로 평가되었다. 사용자의 권한 부여시는 다른 테스팅 도구들과 테스팅 자동화 프레임워크 간에는 차이가 없음을 알 수가 있다.

표 5.2 권한 부여와 접근제어 기능에 대한 테스팅 결과
Table 5.2 Testing Result of Authorization and Access Control

테스트 내용 \ 결과		테스팅 도구						테스팅 자동화 프레임워크					
		R 제품		A 제품		C 제품		R 제품		A 제품		C 제품	
		접근 제어	권한 부여	접근 제어	권한 부여	접근 제어	권한 부여	접근 제어	권한 부여	접근 제어	권한 부여	접근 제어	권한 부여
아이디 필드	빈 필드	-	o	-	o	-	o	-	o	-	o	-	o
	1 ~ Max-1개 문자	-	o	-	o	-	o	-	o	-	o	-	o
	최대 문자	-	o	-	o	-	o	-	o	-	o	-	o
비밀번호 필드	빈 필드	-	o	-	o	-	o	-	o	-	o	-	o
	1 ~ Max-1개 문자	-	o	-	o	-	o	-	o	-	o	-	o
	최대 문자	-	o	-	o	-	o	-	o	-	o	-	o
로그인 버튼	아이디 또는 비밀번호 누락	o	-	△	-	o	-	o	-	o	-	o	-
	허용안된 아이디/비밀번호	o	-	o	-	△	-	o	-	o	-	o	-
	가능한 아이디/비밀번호	△	-	o	-	o	-	o	-	o	-	o	-

o : 만족, △ : 거의 만족

5.3.2 정확성과 파일 무결성 평가

정확성과 파일 무결성에 대한 실험 결과는 <표 5.3>과 같다. 정확성이란 알고리즘의 실행결과가 동일함을 의미하며, 파일 무결성이란 정의된 데이터 타입 대해서 파일 내용상에 하자가 없음을 의미한다. 즉, 동일한 파일을 접근할 때마다 일관된 데이터가 검색된다는 것을 의미한다.

표 5.3 정확성과 파일 무결성에 대한 테스팅 결과
Table 5.3 Testing Result of Collectress and File Integrity

테스트 단계 \ 결과	테스팅 도구						테스팅 자동화 프레임워크					
	R 제품		A 제품		C 제품		R 제품		A 제품		C 제품	
	정확성 0/X	무결성 0/X	정확성 0/X	무결성 0/X	정확성 0/X	무결성 0/X	정확성 0/X	무결성 0/X	정확성 0/X	무결성 0/X	정확성 0/X	무결성 0/X
로그인	o	o	o	o	o	o	o	o	o	o	o	o
R 제품 테스트 스크립트 생성	o	o	-	-	-	-	o	o	-	-	-	-
테스트정보 저장 시스템에 저장	-	-	-	-	-	-	o	o	o	o	o	o
A 제품 테스트 스크립트 생성	-	-	o	o	-	-	-	-	o	o	-	-
C 제품 테스트 스크립트 생성	-	-	-	-	o	o	-	-	-	-	o	o

o : 만족, △ : 거의 만족

<표 5.3>에 의하면 세 종류의 테스팅 도구들과 테스팅 자동화 프레임워크에서 정확성과 파일 무결성의 실험 결과는 전혀 차이가 없이 순차적으로 잘 실행되었다는 것을 나타내고 있다.

5.3.3 시스템 성능 평가

일반적으로 성능 평가에는 리스크의 가능성과 심각성을 고려하는 리스크 매트릭스를 사용하며, 리스크의 종류는 Worst-Case, Unsure, Good/Not Bad 및 Best -Case로 분류한다.

리스크의 등급을 리스크 내용의 사례에 따라 분류하고, 리스크 내용에 따라 리스크의 발생 확률을 계산하기 위하여 리스크 발생 가능성과 리스크 영향의 심각성을 곱셈한다.

리스크 분석표를 기반으로 성능을 평가하기 위하여 3종류의 테스팅 도구와 테스팅 자동화 프레임워크를 실험한 결과는 <표 5.4>와 같다.

표 5.4 리스크 매트릭스에 의한 성능 평가 결과
Table 5.4 Performance Result of Risk Metric

테스트 등급 \ 내역		테스팅 도구						테스팅 자동화 프레임워크					
		R 제품		A 제품		C 제품		R 제품		A 제품		C 제품	
		CPU	메모리	CPU	메모리	CPU	메모리	CPU	메모리	CPU	메모리	CPU	메모리
Worst-Case (1)	SW1	1.38	10.9	1.01	37.50	14.25	43.50	1.10	9.83	0.80	33.38	11.26	38.72
	SW2	1.08	13.5	1.01	36.45	21.75	43.50	0.86	11.97	0.80	32.81	17.40	39.15
	SW3	1.95	13.9	0.18	37.50	4.50	43.50	1.56	12.57	0.14	33.38	3.60	39.15
Unsure (2)	SW1	1.20	9.46	0.87	32.50	12.35	37.70	0.96	8.52	0.70	28.93	9.76	33.55
	SW2	0.94	11.66	0.87	31.59	18.85	37.70	0.75	10.38	0.70	28.43	15.08	33.93
	SW3	1.69	12.10	0.16	32.50	3.90	37.70	1.35	10.89	0.12	28.93	3.12	33.93
Good/ Not Bad (3)	SW1	1.01	8.01	0.74	27.50	10.45	31.90	0.81	7.21	0.59	24.48	8.26	28.39
	SW2	0.79	9.87	0.74	26.73	15.95	31.90	0.63	8.78	0.59	24.06	12.76	28.71
	SW3	1.43	10.24	0.13	27.50	3.30	31.90	1.14	9.22	0.10	24.48	2.64	28.71
Best -Case (4)	SW1	0.92	7.28	0.67	25.00	9.50	29.00	0.74	6.55	0.54	22.25	7.51	25.81
	SW2	0.72	8.97	0.67	24.30	14.50	29.00	0.58	7.98	0.54	21.87	11.60	26.10
	SW3	1.30	9.31	0.12	25.00	3.00	29.00	1.04	8.38	0.09	22.12	2.40	26.10

평균 CPU 시간(%) : 테스팅 도구 = 4.27, 테스팅 자동화 프레임워크 = 3.41
평균 점유 메모리(%) : 테스팅 도구 = 25.43, 테스팅 자동화 프레임워크 = 22.77

<표 5.4>에 의하면 3종류의 테스팅 도구를 각자 수행했을 때 보다 테스팅 자동화 프레임워크를 이용하여 테스트 케이스 정보를 공유할 경우 CPU 처리시간은 평균 20.3% 이상 감소되며, 메모리 점유율은 평균 10.5% 감소되는 것으로 나타났다.

<표 5.4>의 결과를 보다 객관적으로 증명하기 위하여 테스트 등급 중에서 최상 조건(Best-Case)을 선택하여 세부 실험을 수행한 결과는 <표 5.5>와 같다.

표 5.5 실험용 소프트웨어를 이용한 성능 실험 결과
Table 5.5 Performance Result of Software for Experiment

내역 / 테스트 등급		테스팅 도구						테스팅 자동화 프레임워크					
		R 제품		A 제품		C 제품		R 제품		A 제품		C 제품	
		CPU	메모리	CPU	메모리	CPU	메모리	CPU	메모리	CPU	메모리	CPU	메모리
1개 SW	SW1	0.92	7.28	0.67	25.00	9.50	29.00	0.74	6.55	0.54	22.25	7.51	25.81
	SW2	0.72	8.97	0.67	24.30	14.50	29.00	0.58	7.98	0.54	21.87	11.60	26.10
	SW3	1.30	9.31	0.12	25.00	3.00	29.00	1.04	8.38	0.09	22.25	2.40	26.10
2개 SW 조합	SW1과 SW2	0.65	7.58	0.55	25.00	11.57	29.00	0.51	6.82	0.43	22.00	9.14	25.52
	SW2와 SW3	0.76	8.65	0.11	25.00	2.38	30.00	0.59	7.79	0.09	22.00	1.88	26.40
	SW3과 SW1	1.14	8.38	0.40	26.00	10.00	29.00	0.89	7.46	0.31	23.14	7.90	25.52
3개 SW 조합	SW1과 SW2와 SW3	1.07	8.11	0.51	25.00	7.84	29.00	0.86	7.22	0.40	22.00	6.12	25.52

평균 CPU 시간(%) : 테스팅 도구 = 3.23, 테스팅 자동화 프레임워크 = 2.55
평균 점유 메모리(%) : 테스팅 도구 = 20.81, 테스팅 자동화 프레임워크 = 18.45

실험에는 실험용 소프트웨어 3개(SW1, SW2, SW3)를 임의로 선택하고 조합하여 이용하였다. 선택한 실험용 소프트웨어는 3종류의 홈 페이지를 사용하였고, 성능 평가 실험을 위하여 사용자 로그인한 후 사용자 권한 인증을 수행하였다.

<표 5.5>에 의하면 3종류의 테스팅 도구를 각자 수행했을 때 보다 테스팅 자동화 프레임워크를 이용하여 테스트 케이스 정보를 공유할 경우 테스팅 자동화 프레임워크 상에서 CPU 처리시간은 평균 21.0% 감소되었으며, 메모리 점유율은 평균 11.3% 감소되었다.

결과적으로, 테스팅 자동화 프레임워크는 테스트 케이스 정보의 공유와 재사용이 가능하므로 시스템의 성능을 향상시킬 수 있는 것으로 입증되었다.

5.3.4 실험 결과의 종합 평가

위와 같은 실험 결과들을 종합적으로 분석한다면 다음과 같은 결론을 도출할 수 있다.

첫째, 권한 부여와 접근 제어의 실험 결과로서 각 테스팅 도구와 테스팅 자동화 프레임워크 간에는 성능 면에서 차이점이 없다는 것은 테스팅 자동화 프레임워크의 유효성이 입증되었다는 것을 의미한다.

둘째, 정확성과 파일 무결성 실험 결과로서 각 테스팅 도구와 테스팅 자동화 프레임워크가 테스트 스크립트를 동일한 순서로 정확히 생성하였다는 것은 파일의 무결성을 보장한다는 것을 의미한다.

셋째, 시스템 성능 실험 결과로서 각 테스팅 도구보다 테스팅 자동화 프레임워크를 사용했을 경우 CPU 처리시간은 평균 20.3%, 메모리 점유율은 평균 10.5% 감소되었다는 것은 테스팅 자동화 프레임워크를 이용할 경우 테스트 케이스 정보의 공유와 재사용이 가능하다는 것을 의미한다.

Ⅵ. 결 론

테스팅 자동화 도구를 이용하여 테스트 프로세스의 전체 또는 일부를 자동처리 함으로써 제한된 범위 내에서 테스팅 시간과 비용을 줄일 수 있다. 현재 상용화된 테스팅 자동화 도구들은 상호 호환성을 고려하지 않고 개발되었기 때문에 테스트 수행 단계에서 생성된 테스트 케이스 정보의 공유와 재사용이 불가능하다.

본 논문에서는 소프트웨어의 테스팅 시간과 비용을 감소시키기 위하여 테스트 수행 단계에서 생성된 테스트 케이스 정보를 다른 테스팅 도구들이 접근, 이용할 수 있는 소프트웨어 테스팅 자동화 프레임워크를 제안하였다. 제안한 테스팅 자동화 프레임워크의 유효성과 효과성을 입증하기 위하여 기존 3개의 테스팅 자동화 도구들을 테스트 시나리오에 따라 실험하였다. 실험 결과로서, 접근 제어와 권한 부여 평가, 그리고 정확성과 파일 무결성 평가에서 테스팅 자동화 프레임워크의 유효성과 효과성이 3개의 테스팅 자동화 도구들에서 동일하게 나타났다. 또한 시스템 성능 평가에서 테스팅 자동화 프레임워크를 이용할 경우 CPU 처리시간은 평균 20.3% 이상 감소되었으며, 메모리 점유양은 평균 10.5% 감소된 것으로 분석되었다.

결과적으로, 소프트웨어 테스팅 자동화 프레임워크를 이용할 경우 다른 테스팅 자동화 도구들로부터 테스트 케이스 정보의 공유와 재사용이 가능하므로 소프트웨어 테스팅 시간과 비용을 감소시킬 수 있다는 것이 증명되었다.

본 연구에서는 웹 기반 시스템 테스팅을 위해 테스트 수행단계에서 테스트 항목과 리스크 항목을 사전에 정하여 리스크 매트릭스를 구성하고 이용하였다.

향후 소프트웨어 개발주기의 각 단계에서 생성되는 테스트 케이스 정보의 공유가 가능하며, 테스팅 결과에 따라 이러한 정보를 재활용할 수 있는 테스팅 자동화 프레임워크 설계에 대한 연구가 수행되어야 할 것이다.

참고문헌

[1] G. J. Myers, The Art of Software Testing, John Willy & Sons, 1979

[2] IEEE Standard Glossary of Software Engineering Terms, IEEE Society Press, Addison Wesley, 1993

[3] E. Dustin, J. Rashka and J. Paul, Automated Software Testing, Addison Wesley, 1999

[4] M. Fewster, D. Graham and Software Test Automation: Effective use of test execution tools, ACM Press, Addison Wesley, 1999

[5] 정창신, 정순기, "소프트웨어 자동 테스팅 도구의 발전 로드맵 분석", 한국컴퓨터정보학회논문지, 제9권 제1호, pp. 17-23, 2004.3

[6] The Open Group, TETware White Paper, March 2003

[7] OASIS, ebXML Test Framework Committee Specification V1.0, March 2003

[8] M. R. Blackburn, R. Busser and A. Nauman, "Eliminating Requirement Defects and Automating Test," Software Productivity Consortium, 2001

[9] W. E. Perry, Effective Methods for Software Testing, John Wiley & Sons, 2000

[10] 김길준, "XML과 관계형 데이터베이스 매핑을 통한 자료의 변환", 한국컴퓨터정보학회논문지, 제9권 제4호, pp. 5-12, 2004.12

[11] S. H. Kan, Metrics and Models in Software Quality Engineering, Addison-Wesley, 2002

저 자 소 개

정 창 신(Chang-shin Chung)

2004년 8월 : 충북대학교 컴퓨터 공학과 박사

1984년 3월~2001년 11월 : ETRI 팀장/책임연구원

2001년 12월 ~현재 : TTA S/W시험인증센터 실장/책임연구원

<관심분야> S/W제품품질평가, S/W프로세스개선, 데이터베이스 시스템, 통신 소프트웨어

이 계 임(Kye-Im Lee)

2004년 2월 : 충북대학교 컴퓨터공학과 박사수료

1995년 3월~1999년 2월 : 서울가능대학교 정보기술과 조교수

1999년 3월~현재: 인천기능대학교 컴퓨터정보과 부교수

<관심분야> 데이터베이스 시스템, 멀티미디어 시스템

김 종 희(Jong-Hee Kim)

2004년 2월 : 충북대학교 컴퓨터공학과 박사수료

2001년 3월~2004년 6월 : 호서대학교, 선문대학교, 단국대학교, 남서울대학교 시간강사

2004년 9월~현재 : 선문대학교 강의전담교수

<관심분야> 데이터베이스 시스템, 멀티미디어 시스템

정 순 기(Soon-Key Jung)

1982년 8월 : Uni. of Dortmund, Informatik, Dipl. Inform 취득

1994년 2월 : Uni. of Groningen, Computing Science, Dr. 취득

1985년 5월 ~현재 : 충북대학교 컴퓨터공학과 교수

1994년 8월 : 충북대학교 전자계산소장

1998년 11월 : 한국과학재단 한독기초과학협력위원회 정보분과위원장

2000년 4월 : 충북대학교 도서관장

<관심분야> 데이터베이스 시스템, 소프트웨어공학, 소프트 실시간 시스템