EJB 컴포넌트 시험 평가 체계 및 시험 환경 구축에 관한 연구

한국과학기술원 오승욱 · 마유승 · 배두환* · 권용래*
한국전자통신연구원 김길조 · 구자경

1. 서 론

컴포넌트 기술은 소프트웨어를 신속하고 효과적으로 개발할 수 있는 대안으로 90년대 초반부터 각광을 받기 시작하였다. 컴포넌트 기반 기술을 사용하여 어플리케이션을 개발할 경우에는 컴포넌트 기반 기술이 약속하고 있는 재사용성을 기반으로 어플리케이션 개발의 생산성을 높일 수 있어서, 기하급수적으로 증가하고 있는 소프트웨어의 수요를 충당할 수 있게 된다. 따라서 컴포넌트 기술을 활용하는 컴포넌트 기반 개발 방법론이라는 새로운 패러다임이 미래 소프트웨어 산업에 있어서의 새로운 대안으로 인식되며 많은 사람들의 관심을 모으고 있다.

컴포넌트 기술은 규격화되고 검증된 소프트웨어 컴포넌트들의 획득과 조립을 통해 소프트웨어 시스템을 개발하는 기술이다. 소프트웨어 컴포넌트는 재사용성과 상호 운용성이 뛰어난 소프트웨어 부품으로서 어플리케이션 개발자는 마치 하드웨어를 조립하듯이 원하는 기능이나 성능을 가진 소프트웨어 컴포넌트를 선택하여 컴포넌트 시스템을 구성할 수 있게 된다. 이렇게 개발된 컴포넌트 기반 시스템은 일부의 부품을 개선된 부품으로 대체함으로써 품질이나 성능을 개선할 수도 있고 새로운 운용환경에 쉽게 이식할 수도 있다. 따라서 기술 변동에 능동적으로 대응할 수 있으며 타 시스템과의 상호 운용성도 보장된다. 이와 같이 컴포넌트 기반의 소프트웨어 개발은 다양한 측면에서 많은 이점을 제공해 주기 때문에 선진국뿐 아니라 국내에서도 빠르게 성장하고 있는 분야이다.

컴포넌트 기반 소프트웨어 시스템의 품질은 그 시스템을 구성하고 있는 각각의 소프트웨어 컴포넌트의 품질에 의존적일 수밖에 없다. 즉 고품질 컴포넌트 기반 시스템을 개발하기 위해서는 이를 이루는 부품에 해당하는 컴포넌트의 품질을 엄격하게 관리하여야 한다. 따라서, 컴포넌트의 기능과 성능이 우수한지에 대해 검증하기 위한 여러 가지 방법들이 적용될 수 있고, 그 가운데 하나의 방법으로 컴포넌트 시험(Testing)을 들 수가 있다. 하지만 많은 연구에 의해 컴포넌트 소프트웨어는 기존의 소프트웨어에 비해 시험에 있어서 여러 가지 어려움이 있을 것으로 예측되고 있다 [1,2,3]. 이러한 문제점들은 주로 독립적으로 개발된 컴포넌트의 획득과 조립으로 이루어지는 컴포넌트 기반 시스템 개발 방법의 근본적인 속성에서 기인한다. 우선, 컴포넌트 기반 시스템의 부속품으로써 사용된 컴포넌트는 조립되기 전에는 시스템에 사용된 다른 부품들과 함께 시험되기 어렵다. 그래서 컴포넌트 개발자들은 컴포넌트가 앞으로 어떻게 사용될지를 모두 예측하여 시험해 보아야 하는데, 컴포넌트의 모든 행위를 완벽하게 시험하는 것은 불가능하다[1]. 또한, 대부분의 시험 기법들이 코드 기반이라는 점도 문제점으로 지적할 수 있다. 특히 시험이 품질 평가의 수단으로 사용될 경우에는 컴포넌트의 개발과 무관한 제삼자 기관에서 시험이 이루어지는 것이 일반적이기 때문에 대부분의 경우 시험자는 컴포넌트의 소스 코드에 접근할 수 없게 된다. 따라서 컴포넌트의 시험을 위해서는 추가의 정보

* 종신회원

가 컴포넌트 생산자로부터 주어져야한다[1,2,3].

본 논문에서는 제삼자 컴포넌트 시험 평가 체계를 제안하여 앞서 언급한 문제점을 해결하려 한다. 시험 평가 체계는 크게 1) 개발자에 의한 사전 시험 단계와 2) 제삼자 시험 기관에 의한 시험 평가 단계로 이루어진다. 개발자에 의한 사전 시험 단계에서는 개발자가 수행하는 시험을 체계적으로 정리하여 추후에 제삼자 시험 기관의 시험 평가에 활용할 수 있도록 유도하고, 제삼자 시험 기관에 의한 시험 평가 단계에서는 개발자로부터 넘겨받은 시험 자료를 검증하여 시험이 충분히 수행되었는지 평가하고, 필요하다면 추가 시험을 수행함으로써 컴포넌트를 시험 평가할 수 있도록 하고 있다. 하지만 적당한 기준이 없다면 일반적으로 컴포넌트가 충분히 시험되었는지를 판단하는 것은 어려운 일이다[1,4]. 본 연구에서는 동치 영역 분할 기법[6]을 컴포넌트의 입력 도메인에 적용하고 분석된 결과를 토대로 시험 자료를 평가하도록 하고 있다. 본 연구에서는 동치 영역 분할 기법은 컴포넌트의 인터페이스 메소드에 적용된다. 컴포넌트 소프트웨어의 기능성은 인터페이스를 통해 표현된다[5]고 볼 수 있으므로, 인터페이스 메소드의 입력으로 사용되는 파라미터 값들의 조합을 입력 도메인으로 간주하고 동치 영역 분할 기법을 적용하고 있다. 동치 영역 분할의 결과는 시험 자료의 생성과 평가에 모두 사용되어 전체 시험 평가 체계의 건전성을 보장하는 장치로 활용되고 있다.

또한 본 연구에서는 시험 공정의 효율성을 높이기 위해 자동화된 컴포넌트 시험 환경을 제공하고 있다. 우리는 EJB 컴포넌트 시험 도구인 컴포넌트 시험 관리자(CTM : Component Testing Manager)를 구현하였다. CTM은 시험이 적용된 EJB 빈들로 시험 단위를 만들고, 도구에 맞게 고안된 시험 스크립트로부터 시험 사례를 읽어 들이고, 시험 사례로부터 시험 드라이버를 생성하고, 시험 사례를 수행하여 결과를 비교하며, 마지막으로 시험 결과에 대한 간단한 요약서를 생성한다. CTM은 시험의 전체 공정을 지원하지는 못하지만, 시험 수행과 시험 결과 분석의 과정을 일부 자동화하고 있다. 많은 시험 사례를 도구의 지원 없이 직접 수행해보고 결과를 분석하는 작업은 많은 시간을 필요로 하는 작업일뿐 아니라, 오류를 범하기도 쉽다. CTM을 사용할 경우 시험에 드는 노력과 시간뿐 아니라 오류를 범할 가능성도 줄일 수 있을 것이다.

본 논문은 다음과 같이 구성된다. 2절에서는 컴포넌트 시험과 관련된 기존 연구들과 최근 동향에 대해 살펴보고, 3절에서는 제삼자에 의한 컴포넌트 시험 체계에 대해 설명하겠다. 그리고 4절에서는 EJB 컴포넌트에 특화된 시험 자동화 도구의 구현을 토대로 컴포넌트 시험 환경에 대해 언급하고, 마지막으로 5절에서 결론 및 향후 연구 방향을 제시하기로 하겠다.

2. 관련연구

컴포넌트 기반 소프트웨어 개발 기술은 고품질의 소프트웨어를 효율적으로 만들 수 있다고 약속해 왔다. 하지만, 컴포넌트 기반 소프트웨어의 품질은 그 소프트웨어를 이루고 있는 컴포넌트의 품질에 의존적일 수밖에 없고, 따라서 컴포넌트의 품질을 보장하기 위해 시험 등의 방법을 적용할 필요가 있다.

컴포넌트의 시험은 컴포넌트의 성격상 기존의 소프트웨어 시험과 차이를 보이게 된다. Weyuker는 [1]에서 컴포넌트가 사용되는 다양한 환경에 따라 컴포넌트가 매번 재시험되어야 한다는 점을 지적하고 있다. ARIANE 5의 경우 동일한 컴포넌트를 다른 환경에서 사용할 경우 모든 환경에서의 쓰임새를 충분히 예측하고 시험해 보는 것이 어렵기 때문에 새로운 환경에서의 시험이 반드시 필요했음에도 불구하고 시험을 소홀히 했기 때문에 사고가 발생했다는 점에서 컴포넌트의 재사용시의 추가 시험이 반드시 필요하다는 점을 지적하고 있는 것이다. 또한 컴포넌트 소프트웨어의 성격상 소스코드를 접근할 수 없는 입장에서 시험을 수행해야 할 경우가 있기 때문에 기존의 소프트웨어에 많이 적용되는 시험 기법을 적용할 수 없게 된다는 것을 지적하고 있다[1,2,3]. 위와 같은 문제점들을 해결하기 위해서는 기존의 소프트웨어 시험과는 다른 각도의 접근이 필요하고 경우에 따라서는 컴포넌트의 시험을 위해 추가적인 정보가 제공되어야 한다[1,2,3]. Richardson은 회상자(Retrospector)라는 소프트웨어 개체를 추가하여 컴포넌트의 시험이나 운영 경력을 기록하게 하고 이를 재시험에 활용할 수 있도록 제안하

고 있고[2], Harrold 등도 컴포넌트에 대한 요약 정보를 통해 컴포넌트 소스코드에 대한 접근 없이도 컴포넌트를 시험할 수 있도록 컴포넌트 개발자의 지원이 필요함을 지적하고 있다[3].

컴포넌트의 시험이 기존의 소프트웨어 시험과 차이를 보이기 때문에 컴포넌트 시험에 있어서 적용되는 시험 기준(testing criteria) 또한 차이를 보이게 된다. Resenblum은 [4]에서 컴포넌트의 정형화된 모델을 정의하여 컴포넌트 시험의 타당한 기준(adequacy criteria)을 정의하려고 하였고, Ghosh 등은 [5]에서 컴포넌트 인터페이스의 뮤테이션(mutation)을 통해 컴포넌트 시험의 기준을 마련하고자 하였다. 본 논문에서는 컴포넌트의 인터페이스에 대한 도메인 분석 결과인 동치 영역 분석표를 활용하여 이 문제를 해결하고 있다.

또한 컴포넌트의 통합 시험에 있어서는 컴포넌트 사이의 상호 작용과 연관 관계를 표현하고 있는 아키텍처 명세를 시험에 활용하고자 하는 연구들이 진행되었다[8,9,10]. 이들 연구에서는 컴포넌트 사이에 주고받는 데이터나 연관관계 등을 기준으로 시험을 수행할 수 있는 근거를 제시하고 있다.

3. 시험 평가 체계

본 연구에서는 컴포넌트 시험 평가 체계를 크게 개발자에 의한 시험 평가 준비와 제삼자 시험 평가 기관에 의한 시험 평가 수행의 2단계로 구성하였으며 각 단계는 아래와 같이 세부 작업으로 나뉘어 진다.

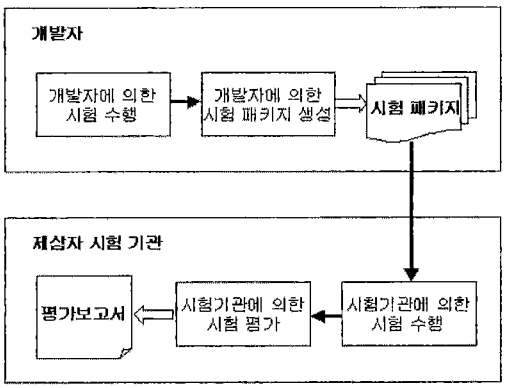


그림 1 컴포넌트 시험 평가 체계

- 개발자에 의한 시험 평가 준비
  - 개발자에 의한 사전 시험 수행
  - 개발자에 의한 시험 패키지 생성
- 제삼자 시험평가 기관에 의한 시험 평가 수행
  - 제삼자 시험평가 기관에 의한 시험 수행
  - 제삼자 시험평가 기관에 의한 평가

그림 1은 컴포넌트 시험 평가 체계를 나타낸다. 개발자는 개발자에 의한 시험 평가 준비 단계의 결과물로 시험 패키지를 제삼자 시험평가 기관에 제공해야 하며 제삼자 시험평가 기관에서는 개발자가 제공한 시험 패키지를 이용하여 시험 평가를 수행한 뒤 평가 결과를 보고한다.

3.1 개발자 시험 평가 준비

개발자 시험 평가 준비 단계는 크게 개발자에 의한 사전 시험 수행과 개발자에 의한 시험패키지 생성의 2단계로 나뉘어지며 이 단계의 결과물로 개발자는 시험패키지를 제삼자 시험기관에 제공해야 한다.

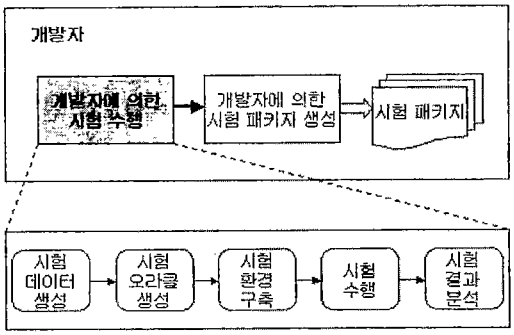


그림 2 개발자에 의한 사전 시험

3.1.1 개발자 사전 시험 수행

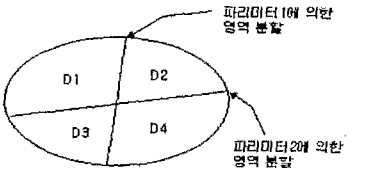
개발자 사전 시험은 컴포넌트 시험 평가에 앞서 개발자가 자체적으로 시험을 수행하는 작업으로서 이 작업을 통해 개발자는 컴포넌트의 품질을 자체적으로 살펴 볼 수 있게 되고 시험 수행 과정에서 발생하는 산출물들은 "개발자에 의한 시험 패키지 생성"에 사용된다.

개발자 사전 시험은 개발자의 시험평가 준비 작업 중 맨 처음 수행하게 되는 작업으로 그림 2의 단계 중 어둡게 보이는 것이 개발자에 의한 사전 시험 수행 단계에 해당되며 다시 세부적으

로 시험데이터 생성, 시험 오라클 생성, 시험환경 구축, 시험 수행, 시험 결과 분석 과정으로 나뉜다.

본 연구에서는 개발자 사전 시험 기법으로 제삼자 시험기관에서 사용될 시험기법 중 기능성에 초점을 둔 시험기법을 제안한다. 본 연구에서는 기능성에 초점을 둔 시험기법의 하나로 동치영역분할(Equivalent Partitioning)에 기반한 기법을 예시하고 있다. 이 기법은 일반적으로 시험 대상 소프트웨어에 대한 정형적인 명세가 부족한 경우에도 적용이 가능한 기법으로서 기능 위주의 단위 시험이나 통합 시험에 널리 사용되고 있다. 이 기법에서는 시험 데이터(Test Data)의 생성과 시험 오라클(Test Oracle) 작성에 동치영역분석표(Equivalent Domain Analysis Table)를 이용한다. 동치영역분석표는 각 메소드(Method)에 대해서 메소드 파라미터(Method Parameter)값과 컴포넌트 상태(State)들을 도메인으로 간주하고 도메인을 같은 기능을 수행하는 동치 영역으로 나눈 정보를 기재한 표이다. 그림 3은 동치영역분할과 동치영역분석표의 예를 보여준다.

그림 3은 인터페이스 메소드의 두 개의 파라미터를 기준으로 도메인을 네 개의 동치 영역으로 분할하고, 그 결과를 동치영역분석표로 표현하고 있다. 그림 3에서 전제조건은 컴포넌트의특정 상태를 의미하는 것으로 인터페이스 메소드의 파라미터와 함께 동치 영역을 분할하는 기준이 된다. 동치영역 분석표에서 식별된 동치영역들을 이용하여, 컴포넌트 개발자는 해당 동치영역을 시험할 수 있는 시험 데이터를 생성할 수 있고, 생성된 시험 데이터의 시험 결과를 확인할 수 있도록 시험 오라클을 함께 작성하여 제공할 수도 있다. 시험 데이터는 식별된 동치영역마다 적어도 하나씩이 생성되어야 하고, 동치영역을 구성하는 파라미터의 대표값이나 경계값을 시험 데이터로 선택하는 것이 일반적이다. 본 연구에서는 동치영역을 활용한 시험 데이터 생성에 대한 지침서를 제공하여 활용하도록 하였으나, 본 논문에서는 자세히 다루지 않기로 하겠다.



동치영역번호	전제조건	파라미터1	파라미터2	설명
D1	DC	> 0	>= 10	...
D2	DC	<= 0	>= 10	...
D3	DC	> 0	< 10	...
D4	DC	<= 0	< 10	...

그림 3 동치영역분할과 동치영역분석표의 예

표 1 시험 패키지 구성 요소

컴포넌트 설치를 위한 자료	
컴포넌트 설치 설명서	컴포넌트가 설치되기 위한 환경에 대한 정보와 설치를 위한 작업에 관한 설명을 담는다.
컴포넌트	EJB 빈을 이루고 있는 클래스 파일들이 제공된다.
컴포넌트 설명서 및 개발문서	컴포넌트 사용자 지침서, 컴포넌트 기능 설명서 등의 사용자 참고 문서와 함께 컴포넌트 개발 과정에서 생성된 개발 문서들이 포함된다.
컴포넌트 제삼자 시험을 위한 자료	
시험 설명서	시험에 사용되는 시험 스크립트로서, 시험 사례와 시험 사례를 수행하기 위해 필요한 환경을 구성하는 방법과 시험 수행의 방법을 기술한다.
주요 기능 요약서	컴포넌트의 주요 기능을 열거한 문서이다. 열거된 주요 기능은 시험 설명서에 기재된 시험 사례와의 연관 관계가 반드시 명시되어야 한다.
시험 오라클	시험 설명서에 기재된 시험 사례를 수행했을 때 예상되는 출력을 기록하고 있는 자료이다.
도메인에 대한 동치 영역 분석표	개발자의 사전 시험에서 작성된 동치 영역 분석표로서, 식별된 동치 영역과 시험 설명서에 기재된 시험 사례, 시험 오라클의 예상 출력 등과의 연관 관계가 반드시 명시되어야 한다.
시험 드라이버	개발자 시험에 사용된 시험 드라이버가 있다면 제공될 수 있다.
시험 스텁	개발자 시험에 사용된 시험 스텁이 있다면 제공될 수 있다.

3.1.2 개발자 시험 패키지 생성

개발자 시험 패키지는 제삼자 시험평가 기관이 컴포넌트에 대한 시험 평가를 위해 컴포넌트 개발자에게 요구하는 사항들을 기술한 것이다. 시험 패키지는 제삼자 시험을 돕기 위해 개발자에 의해 작성되어 제삼자 시험 기관에 제출된다. 시험 패키지의 내용은 크게 컴포넌트 설치를 위한 자료와 제삼자 시험을 위한 자료로 나눌 수 있다. 컴포넌트 설치를 위한 자료에서는 개발한 컴포넌트를 제삼자 시험 기관에서 사용할 수 있도록 하기 위한 정보들을 담고 있다. 제삼자 시험 기관의 시험자는 이 정보를 참고하여 독립적으로 컴포넌트를 운용할 수 있는 환경을 갖추고 시험을 준비하게 된다. 제삼자 시험을 위한 자료는 주로 개발자의 시험 수행 과정에서 발생하는 자료를 체계적으로 정리하여 제삼자 시험 기관의 시험자로 하여금 컴포넌트의 시험 평가에 사용할 수 있도록 하기 위한 내용을 담고 있다. 표 1은 시험 패키지에 포함될 내용들을 담고 있다. 본 연구에서는 "개발자를 위한 시험 패키지 생성 지침"을 통해 시험 패키지를 생성하는 방법을 다루고 있지만, 자세한 내용은 생략하기로 한다.

3.2 제삼자 시험 기관에 의한 시험 평가 수행

제삼자 시험 기관은 개발자가 제공하는 시험 패키지를 이용하여 시험평가를 수행한다. 제삼자 시험 기관의 시험평가는 크게 시험 기관에 의한 시험 수행과 시험 기관에 의한 시험 평가의 두 단계로 이루어지며 시험 수행 단계는 개발자 사전 시험 결과 분석과 추가시험 수행의 두 단계로 이루어진다. 그림 4는 제삼자 시험 기관에 의한 시험 평가 절차를 보여준다.

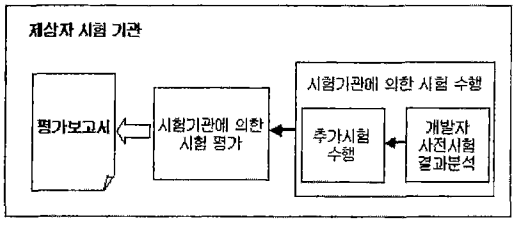


그림 4 제삼자 시험기관의 시험 평가 절차

3.2.1 시험 기관에 의한 시험 수행

시험 기관에 의한 시험 수행은 그림 4와 같이 개발자 사전 시험 결과 분석과 추가시험 수행으로 나눌 수 있다. 각각의 단계에서 수행될 내용을 살펴보면 다음과 같다.

- 개발자 사전 시험 결과 분석

개발자 사전 시험 결과물의 시험 데이터가 적절한지를 조사한 뒤 추가적으로 시험해야 할 사항이 무엇인지 점검한다. 이를 위해서는 개발자가 제공하는 동치 영역 분석표를 활용한다. 우선 시험 데이터가 동치 영역 분석표에서 식별된 동치영역 중 어떤 영역에 해당하는지 확인하고, 식별된 동치 영역 중에 해당하는 시험 데이터가 존재하지 않는 영역이 있는지 검사한다. 시험 데이터에 대한 건전성이 확인된 후에는 시험 데이터를 재사용하여 시험을 수행하고, 개발자의 시험 결과를 확인한다.

- 추가시험 수행

개발자 사전 시험 결과 분석에서 시험되지 않은 것으로 나타난 동치 영역이 있거나, 충분한 시험이 이루어졌다고 판단되지 않을 경우 추가로 시험 데이터를 생성하여 시험을 수행한다. 추가 시험이 필요한 경우는 식별된 동치 영역에 대한 시험 사례가 존재하지 않는 경우나 동치 영역의 분할이 충분하지 못한 경우, 그리고 시험 패키지에서 제공하는 주요 기능 요약서에 나타나는 주요 기능이 시험되지 않은 경우 등을 들 수 있다.

3.2.2 시험 기관에 의한 시험 평가

개발자 사전 시험 결과 분석과 제삼자 추가시험을 통해 얻은 시험 결과를 종합하여 컴포넌트 시험의 평가 결과를 작성한다. 시험의 결과는 수행된 시험 사례에 대하여 시험의 성공 여부로 표현될 수 있고, 이들을 모두 종합한 것이 컴포넌트에 대한 최종 시험 평가 결과가 될 것이다.

이 절에서는 시험 기관에서 수행되는 시험을 크게 두 단계의 절차로 나누어 생각해 보았다. 하지만 시험을 위해서는 이밖에도 시험을 수행하기 위한 시험 환경을 구축하는 작업이 필요하다. 4절에서는 EJB 컴포넌트에 대하여, 시험 수행을 위해 필요한 시험 환경을 살펴보고 자동화 도구를 통해 시험 수행을 지원하려는 노력에 대해 설명하겠다.

4. EJB 컴포넌트 시험 환경

4.1 시험 수행 환경

EJB 시스템은 논리적으로 다음의 3단계(3-tier)로 이루어져 있다[12].

- 클라이언트
- EJB 서버
- 데이터베이스나 그 밖의 영구 저장 매체

여기서는 각각의 구성 요소들은 물리적으로는 서로 다른 기계에 존재할 수도 있고, 아닐 수도 있다는 의미에서 "논리적"이라는 표현을 사용하였다. 3단계 구조에서 클라이언트 측의 어플리케이션은 EJB 서버에 존재하는 EJB 컴포넌트 중에 자신이 원하는 서비스를 가진 컴포넌트를 찾고 이를 통해 자신이 원하는 서비스를 호출할 수 있어야 한다. 3단계 구조를 가정할 경우 EJB 컴포넌트의 수행을 위해서는 그림 5와 같은 구조의 수행 환경을 갖추어야 한다.

그림 5에서 볼 수 있는 것처럼 EJB 컴포넌트를 실행하여 그 기능 및 성능을 시험하기 위해서는 EJB 컴포넌트가 어플리케이션 서버에 설치되어 수행이 되고, EJB 컴포넌트의 수행을 강제하기 위한 시험 드라이버(test driver), 시험 스텁(test stub) 등이 함께 운용되어야 한다. 이 밖에 컴포넌트 시험 수행을 위해서 시험을 수행할 데이터인 시험 사례와 시험 결과를 검토하기 위한 오라클(Oracle) 등이 시험 환경의 입력으로 주어진다.

그림 5에 나타난 개체들은 EJB 컴포넌트의 3단계 구조를 구성하는 EJB 클라이언트, EJB 서버, 데이터베이스, 그리고 시험의 대상이 되는 EJB 컴포넌트 와 시험 스텁으로 구분할 수 있다. 본 논문에서는 EJB 컴포넌트를 EJB 빈과 자바 클래스들의 집합으로 간주하고 있다. 이러한 가정은 현재 구현되고 있는 EJB 컴포넌트들에서 일반적으로 찾아볼 수 있는 것으로 시험을 수행할 대상을 명확히 하기 위하여 필요하다. EJB 컴포넌트에 속하는 EJB 빈들은 시험하려는 기능에 직접적으로 연관되어 있는 것으로 EJB 서버에 설치되어 있어야 한다. 그 밖의 EJB 빈이나 자바 클래스들은 독립적으로 존재하여 시험 대상인 EJB 빈들과 상호 작용을 하며 컴포넌트의 행위를 완성하게 되는데 이들을 시험 스텁(Test Stub)으로 볼 수 있다. 시험 드라이버(Test Driver)는 EJB 빈의 클라이언트로서 자바 어플리케이션이나, 서블릿, 애플릿, 혹은 또 다른 EJB 빈으로 구현될 수 있다. 시험 드라이버는 시험 사례를 입력으로 받아 강제로 EJB 빈을 구동하여 시험 사례를 수행할 수 있도록 하는 역할을 하고 있다. 본 연구에서는 시험 스크립트를 통해 시험 사례를 읽어들여 시험 드라이버를 자동으로 생성하도록 하고 있다.

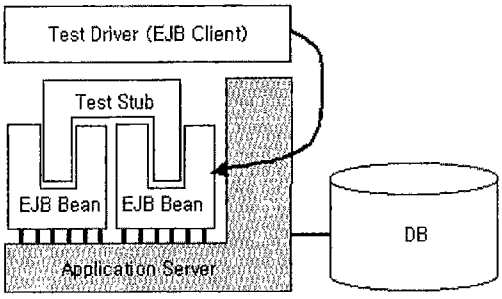


그림 5 EJB 컴포넌트의 시험 수행 환경

4.2 시험 자동화 도구의 구현

본 연구에서는 EJB 빈의 체계적이고 효율적인 시험을 위하여 시험 데이터를 작성하고 수행하며 시험의 결과를 비교하는 일련의 과정을 자동화하여 시험자의 노력을 덜어주는 것을 목적으로 하는 도구인 컴포넌트 시험 관리자 (CTM: Component Test Manager)를 개발하였다. CTM(Component Test Manger)은 컴포넌트 시험의 공정 중 일부분을 자동화한 도구로서 시험자가 시험을 수행할 경우 시험의 효율성을 높이며 도구에서 제공하는 체계적인 시험 공정을 따라가며 시험을 수행함으로써 보다 신뢰성 있는 시험을 수행할 수 있도록 도와준다.

시험은 시험할 데이터의 개발과 개발된 데이터의 수행을 통해 이루어진다. 따라서 시험 사례의 개발과 수행 등에 관련된 작업은 시험의 주를 이루는 작업이고, 시험 공정을 시험 사례(test case)의 생명 주기에 따라 다음과 같이 구분할 수 있다[11].

식별(Identify) 무엇을 시험할 것인지 정하고 우

선순위를 매긴다.

개발(Design) 시험 데이터를 만들 방법을 정하고 시험 데이터를 설계한다.

작성(Build) 시험 데이터를 수행할 수 있는 형태로 구현한다.

수행(Execute) 시험 데이터를 수행한다.

비교(Compare) 시험 데이터의 수행 결과와 예상 결과를 비교한다.

앞서 열거한 공정 중에서 CTM 에서는 식별, 개발 이후의 과정인 작성, 수행, 비교의 과정을 자동화하고 있다. 즉 CTM은 시험 데이터를 시험할 대상인 프로그램에 쉽게 적용할 수 있도록 도와주는 자동화 시험 도구로서 의미를 지닌다. 따라서 대량의 시험 데이터의 효율적인 처리가 가능하도록 시험자를 도울 수 있다. 시험 공정별로 CTM에서 수행하고 있는 기능은 다음과 같다.

• 작성

시험 사례와 시험 사례의 수행 결과, 예상 결과 등을 기록하고 있는 시험 스크립트(test script)를 생성할 수 있는 기능을 제공한다. 또한 이미 제작된 시험 스크립트를 입력으로 받아들여 재사용할 수 있는 기능도 제공한다.

• 수행

시험 스크립트에서 기술하고 있는 내용을 입력으로 시험 드라이버를 자바 어플리케이션 형태로 생성하고 이를 실행시키는 기능을 제공한다.

• 비교

시험 스크립트에서 기술하고 있는 예상 결과를 시험 사례의 수행 결과와 비교하여 차이점을 보고하는 기능을 제공한다.

그밖에 시험 데이터들의 효율적인 관리를 위하여 시험 대상 단위의 시험 환경을 구성하고 이를 기본단위로 시험을 관리하기 위한 전처리기(pre-processor) 를 제공하고 있다. 이를 모두 포함하는 CTM의 전체 구성도는 그림 6과 같다. 그림 6에서 볼 수 있듯이 CTM은 파이프 라인 (Pipe-Line) 형태의 구조를 가지고 있으며 각 부분을 이루는 모듈의 명칭과 역할은 다음과 같다.

• 전처리기(Pre-Processor)

시험이 수행될 단위를 계층적으로 선언하여, 시험이 선언된 단위별로 이루어지고 결과가 집계되도록 한다. 또한 시험 단위에 포함되는 시험 대상 컴포넌트도 지정할 수 있다.

• 시험 사례 작성기(Test Case Builder)

시험 사례로 사용되는 EJB 빈의 인터페이스 호출을 시험자가 직접 기술할 수 있도록 한다.

• 시험 드라이버 생성기(Test Driver Generator)

시험 사례를 입력으로 받아 시험 사례를 수행할 수 있는 시험 드라이버를 생성한다.

• 시험 결과 비교기(Test Outcome Comparator)

시험 결과를 시험 오라클을 사용하여 검사한다.

• 시험 보고기(Test Reporter)

시험 결과 비교의 결과를 시험 단위별로 집계

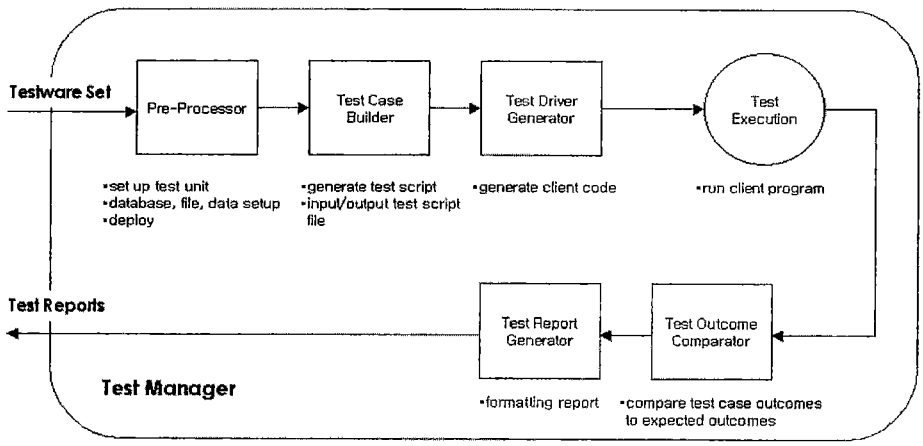


그림 6 컴포넌트 시험 관리자 도구 설계

하여 간단한 결과 요약을 생성한다.

CTM은 EJB 컴포넌트의 시험에 있어서 시험 공정의 후반부에 해당하는 시험 수행과 결과 분석 공정을 자동화하여 전체 시험 공정을 효율성을 높일 수 있을 것이다.

5. 결론 및 향후 연구 방향

컴포넌트의 보급이 확산됨에 따라 품질을 보장할 수 있는 안정된 컴포넌트 제품의 유통을 위한 제도적 장치가 필요하게 되었다. 이를 위해서는 컴포넌트의 품질을 평가할 수 있는 절차와 기법이 필요하고, 이러한 기법의 하나로 컴포넌트 시험을 생각할 수 있다. 컴포넌트 시험은 컴포넌트가 가지고 있는 기능의 정확성, 견딜 수 있는 부하의 정도 등을 알아보는데 유용하게 사용될 수 있을 것이다.

본 연구에서는 컴포넌트의 품질 평가를 위한 시험 체계를 제시하였다. 컴포넌트가 기존의 소프트웨어에 비해 시험이 어려운 점을 감안하여, 독립된 기관에서 컴포넌트의 시험이 원활하게 이루어지기 위해 필요한 산출물들을 정리하였고, 이들을 생성하고 사용하기 위한 적당한 절차를 제시함으로써 컴포넌트의 시험 체계를 정립하였다. 또한 컴포넌트 시험을 위한 시험 환경을 정의하였고, 자동화된 시험 환경의 구축을 위해 시험 지원 도구를 개발하였다. 본 연구를 통해 개발된 시험 지원 도구인 "컴포넌트 시험 관리자(CTM)"는 시험의 전체 공정을 지원하지는 못하지만 시험의 수행에 관련된 부분을 자동화하고 있다. CTM에서는 지정된 형태의 시험 사례를 입력으로 받거나 생성하여 어플리케이션 서버에 설치되어 있는 빈을 통해 수행시키고 그 결과를 시험 오라클과 비교하여 시험을 평가하는 기능을 제공한다.

자동화된 시험 지원 도구는 시험자의 노력을 줄이는 것을 근본 목적으로 한다. 이런 점에서 CTM은 시험의 전체 공정을 다루고 있지 못하다는 점을 지적할 수 있다. 하지만 이러한 약점에도 불구하고 시험 공정에서 많은 시간이 시험 수행에 할애되며, 가장 많이 반복되는 부분이 시험 수행이라는 점을 감안할 때 CTM의 활용 가치를 인정할 수 있을 것이다. 앞으로도 CTM에 시험 계획, 시험 사례 설계 등의 시험 공정 전반에 걸친 관련 도구를 추가로 개발 보완함으로써 전체 시험 공정을 다루는 보다 완벽한 도구로 발전시켜 나갈 계획이다.

또한 CTM은 현재 기능 시험에 초점을 두고 개발되어 다중 클라이언트의 생성을 통한 성능 시험이나 부하 시험의 기능을 제공하고 있지 못하다. 다른 소프트웨어에서와 달리 컴포넌트, 특히 서버 쪽 컴포넌트들의 경우 성능 시험이나 부하 시험 등이 중요한 품질 평가 방안이 될 수 있을 것이므로 이 부분에 대한 꾸준한 연구가 필요할 것으로 생각된다. 현재로서는 기존의 시험 업체에서 개발한 EJB 빈을 대상으로 한 시험 도구는 많지 않은 상태이고, 그 성능이 검증되지 않아 널리 사용되는 도구는 없으며 아직까지는 도구의 성능을 검증 받는 단계라고 볼 수 있다. 따라서 상용 도구의 품질을 꾸준히 벤치마킹하면서 자체적인 도구의 개발을 모색해 볼 필요가 있다.

참고문헌

[1] E. J. Weyuker, "Testing Component-Based Software: A Cautionary Tale," *IEEE Software*, Sep./Oct., 1998.

[2] C. Liu and D. J. Richardson, "Software Components with Retrospectors," *In Proc. of Int. Workshop on the Role of Software Architecture in Testing and Analysis*, pp. 64~48, June 1998.

[3] M. J. Harrold, et. al., "An Approach to Analyzing and Testing Component-Based Systems," *1st Int. ICSE Workshop on Testing Distributed Component-Based Systems*, Los Angeles, California, USA, 17. May, 1999.

[4] D. S. Rosenblum, "Adequate Testing of Component-Based Software," *TR97-34, Univ. of California, Irvine*, 11 Aug. 1997.

[5] S. Ghosh and A. P. Marthur, "Interface Mutation to Assess the Adequacy of Tests for Components and Systems," *TOOLS (Technology of Object-Oriented Languages & Systems) USA 2000*, Santa Barbara, July 30 ~ Aug 3, 2000.

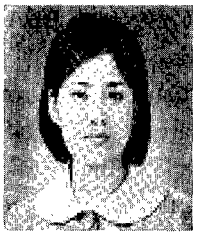
[6] Bingchiang Jeng and Elaine J. Weyuker,

"Some Observations on Partition Testing," *Proceedings of the ACM SIGSOFT '89 third symposium on Software testing, analysis, and verification*, 1989, Pages 38~47.
[7] J. M. Voas, "Certifying Off-The-Shelf Software Components," *IEEE Computer*, June, 1998.
[8] D. J. Richardson and A. L. Wolf, "Software Testing at the Architectural Level," *2nd International Software Architecture Workshop (ISAW-2)*, 1996.
[9] A. Bertolino and P. Inverardi, "Architecture-based Software Testing," *ISAW-2*, 1996.
[10] M. D. Rice and S. B. Seidman, "An Approach to Architectural Analysis and Testing," *ISAW-3*, 1998.
[11] M. Fewster and D. Graham, *Software Test Automation: Effective use of test execution tools*, Addison-Wesley, 1999.
[12] Tom Valesky, *Enterprise JavaBeans*, Addison Wesley, 1999.

오 승 욱

1992~1996 한국과학기술원 전산학과 공학사
1996~1998 한국과학기술원 전산학과 공학석사
1998~현재 한국과학기술원 전자전산학과 전산학전공 박사과정
관심분야:반응 시스템의 요국사항 검증, 정형 기법을 이용한 자동화된 검증 기법, 소프트웨어 시험 기법
E-mail:suoh@salmosa.kaist.ac.kr

마 유 승

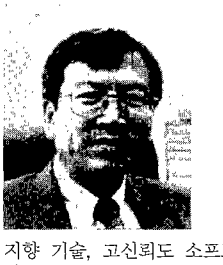


1994~1998 한국과학기술원 전산학과 공학사
1998~2000 한국과학기술원 전산학과 공학석사
2000~현재 한국과학기술원 전자전산학과 박사과정
관심분야:컴포넌트 소프트웨어 시험, Mutation을 이용한 소프트웨어 시험, 소프트웨어 재귀 시험.
E-mail:ysma@salmosa.kaist.ac.kr

배 두 환

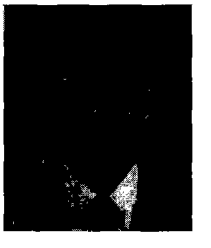
1976~1980 서울대학교 조선공학과 학사
1985~1987 University of Wisconsin-Milwaukee석사
1988~1992 University of Floreda 박사
1992~1994 University of Floreda, Assistant Professor
E-mail:bae@salmosa.kaist.ac.

권 용 래



1969 서울대학교 문리대학 이학사
1971 서울대학교 대학원 이학석사
1971~1974 육군사관학교 전임강사
1978 미국 피츠버그대학 이학박사
1978~1983 미국 Computer Science Corporation 연구원
1983 현재 한국과학기술원 전자전산학과 전산학전공 교수
관심분야:실시간 병렬 소프트웨어 검증, 실시간 시스템의 객체 지향 기술, 고신뢰도 소프트 웨어의 품질 보증, 소프트웨어 시험 기법
E-mail:kwon@salmosa.kaist.ac.kr

김 길 조



1987 서울대학교 산업공학과 졸업 학사
1989 한국과학기술원 경영과학과 졸업 석사
현재 한국전자통신연구원 선임연구원
E-mail:kgj@etri.re.kr

구 자 경

1992.2. 덕성여자대학교 수학과 학사
1994.2. 한국외국어대학교 경영정보대학원 석사
1994.6.~한국전자통신연구원 S/W 공학연구부 선임연구원
E-mail:jkkoo@etri.re.kr