제18회 한국정보처리학회 추계학술발표대회 논문집 제9권 제2호 (2002.11)

에이전트 기반의 컴포넌트 관리 시스템

최신형*, 한판암*, 권대곤**
*경남대학교 정보통신공학부
**남해전문대학 컴퓨터응용정보과
e-mail:cshinh@korea.com

Agent-Based Component Management System

Shin Hyeong Choi*, Pan Am Han*, Dae Gon Kweon**
*Dept of Computer Science and Engineering, Kyungnam University
**Dept of Computer Application, Namhae College

요 약

본 논문에서는 인터페이스를 통해 통신하는 컴포넌트 기반의 분산객체환경에서 컴포넌트 추가와 변경에 따른 변화를 인식하여 테스트하는 에이전트 기반의 테스트 방안을 제시한다.

다중 에이전트로 구성된 컴포넌트 관리 시스템을 이용하면 분산객체환경에서 컴포넌트 변경에 대한 정보를 주기적인 검색을 통해 쉽게 파악할 수 있으며, 변경이 발생하는 컴포넌트에 대해 해당 클라이언트와 서버측 컴포넌트와의 메소드 연결관계정보를 정보DB 내에 저장함으로써 효율적인 컴포넌트 관리가 가능하다. 또한, 이 정보를 바탕으로 불필요하게 모든 컴포넌트에 대한 테스트를 감소시킬 수 있고, 테스트하는데 소요되는 시간과 시스템 부하를 줄일 수 있다.

1. 서론

최근 인터넷의 발달과 웹의 사용이 증가됨에 따라, 컴퓨터 패러다임은 클라이언트 서버 구조에서 분산객체시스템으로 변화하고 있는 추세이며, 실세계에서 발생하는 많은 문제들에 대해 분산환경에서의 서비스 처리가 요구된다.

최근 시스템 규모가 커짐에 따라 컴포넌트 기반 소프트웨어가 현재로서는 대규모 어플리케이션 소프트웨어를 구축하기에 최적의 솔루션이며, 소프트웨어의 품질을 보증하고, 재사용을 통한 소프트웨어 개발 생산성을 향상시키기 위한 방법으로 각광받고 있다. 이와 같은 분산 시스템에서는 다양한 변경 혹은 그에 따른 오류가 발생할 가능성이 더욱더 높아지므로 소프트웨어 신뢰성 향상과 품질 측정을 위한 테스트 작업이 필요하다.

본 논문에서는 인터페이스를 통해 통신하는 컴포넌트 기반의 분산객체환경에서 컴포넌트 추가와 변경에 따른 변화를 인식하여 테스트하는 에이전트 기반의 테스트 방안을 제시한다.

2. 소프트웨어 컴포넌트

컴포넌트란 분산객체환경에서 시스템을 구성하는 기본 단위이다. 소프트웨어 컴포넌트는 잘 알려진 기능을 수행하도록 구현한 단위 소프트웨어로서 완전한 형태로 존재하며, 구체적인 구현은 컴포넌트 소프트웨어의 사용자에게 숨기고 잘 정의된 인터페이스를 통하여 해당 기능을 제공한다. 즉, 하나 혹은 그 이상의 잘 정의된 인터페이스들을 가지는 소프트웨어의 부품이라 할 수 있다[9].

외부에서 컴포넌트에 접근할 수 있는 유일한 방법은 인터페이스를 통하는 방법이며, 컴포넌트도 인터페이스를 거쳐야만 자신의 서비스를 외부에 공개할 수 있다. 그리고 실제 컴포넌트의 구현 부분은 한 개의 객체가 될 필요는 없다. 결국 여러 개의 객체가 하나의 컴포넌트를 이루고 이들 서비스가 인터페이스를 통해 외부에 공개될 수 있다.

3. 에이전트

에이전트는 지속적으로 환경에서 지각된 것과 내부 지식을 바탕으로 추론하며 이를 바탕으로 환경에 영향을 미친다. 또한 사용자를 포함한 다른 에이전트와 지속적으로 의사 소통하는 소프트웨어 요소라고 정의할 수 있다[8]. 현재 많은 수의 컴퓨터 시스템이 자동화된 에이전트를 도입하고 있고, 분산 시스템을 위한 차세대 모델로서도 지지를 받고 있다. 에이전트는 여러 가지 특징이 있지만 Autonomy(자율성), Adaptation(적응성), Cooperation(협력성)이 대표적인 특징이라 할 수 있다.

한편 다중 에이전트 시스템은 에이전트 혼자만의 능력이나 지식으로는 해결하기 힘든 문제를 다른 에이전트들과 상호 협력하여 문제를 풀어나가는 시스템으로 속도, 안정성, 확장성이 용이하다. 이러한 장점은 다중 에이전트 시스템이 문제를 해결하는데 사용하는 조정, 교섭, 통신의 성질을 이용하므로 이뤄질 수 있다[1]. 즉, 개개의 에이전트들이 이루는 집단이 통신을 통하여 서로 화합하고 타협하여 서비스를 제공한다. 따라서, 다중의 에이전트들이 서로 긴밀하게 화합하기 위해서는 이들 사이의 조직적 구조와 이들의 작업과 자원 할당에 대한 적절한 기준이 필요하다. 에이전트간의 통신은 정해진 언어 규약에 따라 메시지를 주고받음을 의미한다. 에이전트는 다른 에이전트에게 서비스를 요청하기 위해 정해진 언어 규약에 따라 요구 사항을 메시지 형태로 바꾼 후 해당 에이전트에게 전달한다. 다른 에이전트로부터 서비스 요청을 받은 에이전트는 그 메시지를 분석해 내부에서 처리할 수 있는 형태로 변환해서 이를 처리한다. 에이전트는 그 결과를 다시 메시지 형태로 바꿔 이를 요청한 에이전트에게 전달한다

4. 컴포넌트 테스트

분산객체환경에서는 각종 어플리케이션이 다수의 분산객체들을 인스턴스화하며, 각각의 객체는 독립적으로 동시에 실행되므로 전통적 분산 프로그램과 같이 비결정적으로 실행된다[7]. 이런 비결정성 때문에 분산 프로그램의 테스트는 순차 프로그램의 테스트보다 난해하다.

분산객체환경에서는 다양한 컴포넌트들이 존재하고 새로운 시스템을 개발하기 위해서는 이전에 만들어진 컴포넌트들을 사용할 수도 있을 것이다. 다른 시스템에서 사용하기 위해서는 시스템에서 요구하는 모습으로 컴포넌트를 수정하는 과정이 필요하며, 이는 컴포넌트의 인터페이스를 대상으로 수행된다.

따라서 컴포넌트를 이용한 소프트웨어 개발과정에서 컴포넌트에 대한 테스트가 선행되지 않을 경우 개발될 시스템에는 치명적인 문제가 발생할 가능성이 항상 존재하므로 컴포넌트 수정 이후에 컴포넌트에 대한 테스트가 반드시 필요하다.

4.1 컴포넌트 관리 시스템

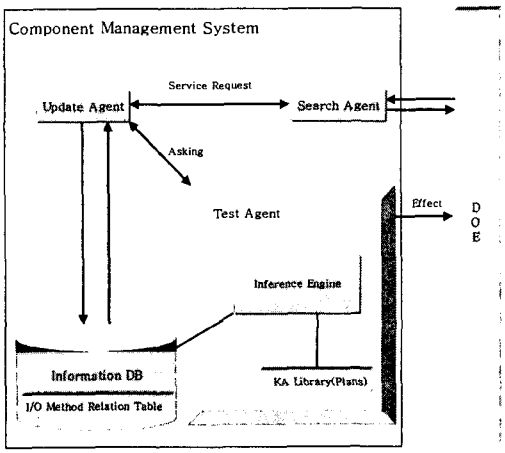
분산객체환경에서는 언제, 어디에서 컴포넌트가 변경 혹은 추가·삭제될 지 예측이 불가능할 뿐 아니라 수많은 컴포넌트를 관리한다는 것은 상당히 어려운 작업이다.

본 논문에서는 상호 작용하는 컴포넌트간 테스트를 위해 컴포넌트 기반의 분산객체환경에서 컴포넌트 추가와 변경에 대해 테스트할 수 있는 컴포넌트 관리 시스템을 설계하며, 이를 이용한 컴포넌트 테스트 방안을 제시한다.

컴포넌트 관리 시스템의 구조는 그림1과 같으며, 검색 에이전트, 갱신 에이전트, 테스트 에이전트, 정보DB로 구성되어진다. 검색 에이전트는 분산객체환경에서 변화, 추가, 삭제된 컴포넌트를 정보DB를 참고하여 검색하며, 갱신 에이전트는 분산객체환경에서 변화된 컴포넌트 정보를 정보DB에 갱신한다. 정보DB는 해당 분산객체환경에서 클라이언트와 서버 관계가 있는 컴포넌트들의 입출력 인터페이스부에 있는 메소드간 대응관계를 테이블 형식으로 저장·관리하고 있다. 테스트 에이전트는 갱신 에이전트를 통해 변경된 정보DB와 KA Library(지식영역 라이브러리)를 바탕으로 추론과정을 거쳐 해당 컴포넌트 테스트를 실시한다. 즉, 테스트 방법은 지식영역라이브러리에 따라 달라질 수 있다: 이때, 각 에이전트간의 통신은 메시지 전달에 의해 수행되며, 해당 메시지는 메시지 타입, 내용과 더불어 메시지를 주고받는 에이전트명으로 구성된다.

이와 같이 컴포넌트 관리 시스템은 검색 에이전트를 사용하여 실시간으로 분산객체환경에서 변화되는 컴포넌트를 관찰하면서 변화가 있는 컴포넌트를 발견하고 이를 정보DB에 있는 입출력 메소드 연결관계 테이블에 기록함으로써 보다 편리한 컴포넌트 테스트 환경을 제공하는 것을 목적으로 한다. 각 에이

전트가 시간이 지남에 따라 해당 분산객체환경에 대한 적응성을 제공하기 위해서는 해당 분산객체환경의 변화되는 컴포넌트 유형이나 특성 등을 학습하여 지식화하고, 이를 효율적으로 이용할 수 있는 기능이 있어야 한다.



[그림1] 컴포넌트 관리 시스템

소프트웨어 컴포넌트들은 입력과 출력 인터페이스들의 연결로 구성되며, 컴포넌트 내부에 있는 객체의 입력 인터페이스는 다른 컴포넌트 내부의 객체들에 의해 호출될 수 있고, 출력 인터페이스는 다른 컴포넌트 내부의 객체 인터페이스를 호출한다.

이와 같이 컴포넌트 관리 시스템 내에 검색, 갱신, 테스트 에이전트와 정보 DB를 포함시킨 다음 분산객체환경에서 컴포넌트들의 추가와 변경이 발생할 때마다 Sensor로부터 받아들인 컴포넌트에 대해 입출력 메소드 연결관계 테이블과의 비교분석과정을 거쳐 테스트 여부를 결정하고, 테스트가 필요한 컴포넌트에 대해서는 지식영역 라이브러리에 명시한대로 테스트를 실시한다. 즉, 매번 입력받은 컴포넌트들을 테스트한다는 것은 시스템 전체에 부하를 많이 주므로 기존에 테스트되고 변경되지 않은 컴포넌트에 대해서는 하나의 필드에 체크를 함으로써 불필요하게 모든 컴포넌트를 테스트하는 것을 줄일 수 있다. 이를 위해 정보DB내에 정의된 입출력 메소드 연결관계 테이블은 표1과 같은 포맷으로 정의되며, 클라이언트 측의 출력 메소드가 특정 서버의 입력 메소드에 연결된다는 것을 레코드 형식으로 나타낸다.

<표 1> 입출력 메소드 연결관계 테이블

i_mod	incoming_ method	target_ server	o_mod	outgoing_ method
0/1			0/1	

5. 적용사례

본 연구에서 제시한 컴포넌트 관리 시스템은 다중 에이전트 구조로서 분산객체환경에서 클라이언트와 서버측 컴포넌트들의 입출력 인터페이스의 메소드 연결관계 정보를 저장하고 있는 정보DB를 바탕으로 컴포넌트들을 테스트할 수 있다.

이 장에서는 이와 같은 테스트 방법을 입출력 인터페이스를 가지는 두 개의 컴포넌트에 적용한 결과를 나타낸다.

두 개의 컴포넌트 A, B는 다음과 같은 구조로 구성된다.

Component A	Component B

Input Interface
interface customer {
string Name();
Name(string Name);
long CustomerID;
};
	Input Interface
	struct strcust {
	store(strcust cust);
	strcust load(long CustID);
	};
	interface StoreCust {
	store(strcust cust);
	strcust load(long CustID);
	};

Output Interface
interface Store{
storeCustomer(Customer cust);
};	

[그림 2] Component A, B 구조

위의 예에서 컴포넌트 A는 입력 인터페이스 customer와 출력 인터페이스 Store를 가지며, 컴포넌트 B는 하나의 입력 인터페이스인 StoreCust를 가짐을 알 수 있다. 여기서 컴포넌트 A의 출력 인터페이스가 컴포넌트 B의 입력 인터페이스에 연결된다면, 컴포넌트 A의 출력 인터페이스의 메소드 Store::storeCustomer(Customer cust)가 컴포넌트 B의 입력 인터페이스 메소드 StoreCust::store (strcust cust)에 연결된다.

이와 같은 연결관계가 성립할 때 다음과 같은 입출력 인터페이스의 메소드 연결관계 테이블이 작성될 수 있다.

<표 2> 입출력 인터페이스 메소드 연결관계정보

i_mod	incoming_ method	target_ server	o_mod	outgoing_ method
0/1	StoreCust::store(strcust cust)	Component B	0/1	Store::storeCustomer(Customer cust)

컴포넌트 관리 시스템의 정보DB에 표2와 같은 정보가 저장되면 수많은 컴포넌트들로 구성된 시스템에서 클라이언트나 서버 측 인터페이스 중 임의의 메소드가 변경될 때 테이블의 해당 레코드에 저장된 i_mod 항목과 o_mod 항목에 수정과 테스트를 수행했다는 표시로 1값을 부여한다.

또한, 클라이언트와 서버 측 컴포넌트들 사이의 입출력 인터페이스 사상관계를 주기적으로 정보DB에 구성해 놓으면 어느 한 쪽의 컴포넌트에 변경이 발생하더라도 대응하는 컴포넌트를 파악할 수 있으므로 별도의 검색을 통해 테스트할 필요없이 i_mod와 o_mod 값에 따라 테스트 여부를 결정함으로써 불필요한 메소드 테스트를 줄일 수 있다.

본 논문에서는 분산객체환경에서 클라이언트의 컴포넌트 요구와 그에 대응하는 서버 측 컴포넌트들의 입출력 메소드 연결관계 테이블을 정보DB내에 정의하여 테스트 에이전트로 하여금 컴포넌트 변경이 발생하는 환경에서 클라이언트와 서버 측의 컴포넌트에 대한 테스트를 실시할 때 연결관계가 있는 컴포넌트들을 함께 테스트할 수 있고, 불필요한 컴포넌트 테스트를 감소시킬 수 있다.

6. 결 론

본 논문에서는 분산객체환경에서 추가와 변경이 반복되는 컴포넌트들을 테스트하기 위해 다중 에이전트로 구성된 컴포넌트 관리 시스템을 설계하고, 입출력 메소드 관계 테이블을 포함하는 정보 DB를 바탕으로 해당 컴포넌트를 테스트하는 방안을 제시하였다.

컴포넌트 관리 시스템은 검색 에이전트, 갱신 에이전트, 테스트 에이전트와 정보DB로 구성되며, 분산객체환경에서 컴포넌트를 검색 에이전트가 검색하여 갱신 에이전트에게 메시지 형태로 전달한다. 갱신 에이전트는 이 정보를 정보DB 내에 저장된 입출력 메소드 연결관계정보와 비교하여 메소드 정보가 변경되었으면 테스트 에이전트가 해당 컴포넌트에 대한 테스트를 실시한다. 즉, 컴포넌트들의 추가와 변경이 발생할 때마다 Sensor로부터 받아들인 컴포넌트에 대해 입출력 메소드 연결관계 테이블과의 비교분석과정을 거쳐 테스트 여부를 결정하고, 테스트가 필요한 컴포넌트에 대해서는 KA Library에 명시한대로 테스트를 실시한다.

본 논문에서 제안한 컴포넌트 관리 시스템을 이용하면 분산객체환경에서 컴포넌트 변경에 대한 정보를 주기적인 검색을 통해 쉽게 파악할 수 있으며, 변경이 발생하는 컴포넌트에 대해 해당 클라이언트와 서버측 컴포넌트와의 메소드 연결관계정보를 정보DB 내에 저장함으로써 불필요하게 모든 컴포넌트에 대한 테스트를 감소시킬 수 있고, 테스트하는데 소요되는 시간과 시스템 부하를 줄일 수 있다.

참고문헌

[1] S. Green, L. Hurst, B. Nangle, P. Cunningham, F. Somer, and R. Evans., *Software Agents : A review*, 1997

[2] H.D. Hofmann, J. Stynes, "Implementation Reuse and Inheritance in Distributed Component Systems", *IEEE Trans, Software*, 1998

[3] W. Kozaczynski and G. Booch, "Component-Based Software Engineering", *IEEE Trans, Software*, 1998

[4] R.J. Nicholas, "On agent-based software engineering", *Artificial Intelligence*, V.117 N.2, 277-296, 1999

[5] H.S. Nwana, *Software Agents: An Overview*, Knowledge Engineering Review, 11(3), pp. 205-244, 1996

[6] S.Y. Park et al, "Agent-based Software Analysis Method in Distributed Environment", *Proceedings of the 1999 IEEE International Fuzzy Systems Conference*, V.1, pp.321-325, 1999

[7] H.W. Sohn, C.K. David and P. Hsia, "State-based Reproducible Testing for CORBA Applications", *Proceedings of the International Symposium on Software Engineering for parallel and Distributed Systems*, pp.24-35, 1999

[8] R. Stuart and N. Peter, *Artificial Intelligence: A Modern Approach*, Prentice-Hall, 1995

[9] S.S. Yau and B. Xia, "Object-Oriented Distributed Component Software Development based on CORBA", *Proceedings of the Twenty-Second Annual International Computer Software & Applications Conference*, pp.246-251, 1998