트리 구조 기반의 컴포넌트 모델 제안

허제민○ 김지홍
경원대학교, 전자계산학과
hjm1980@ku.kyungwon.ac.kr○, wiskjh@kyungwon.ac.kr

Proposing for Component Model Based on Tree Structure

Huh, Je-Min○ Kim, Ji-Hong
Dept. of Computer Science, Kyungwon University

요 약

CBSE(Component-Based Software Engineering)는 현재 많은 연구를 통해 소프트웨어 컴포넌트 모델에 관한 상당한 성과를 이루었다. 하지만 기존의 모델들은 각각이 제안한 프레임웍이 존재하고 그와 관련된 컴포넌트 명세들을 요구한다. 그리고 이를 사용하기 위한 방법을 학습해야한다. 이런 이유로 시장에서 바라는 소프트웨어 컴포넌트의 재사용이 쉽게 이루어지지 않았다. 따라서 컴포넌트의 재사용을 위한 많은 연구들의 공통점들을 연구하여 이상적인 컴포넌트 생명주기가 제안되었고 이를 따르는 모델 또한 제안되었다. 그러나 이 모델은 간접 메시지 전달 방법을 사용한 Exogenous 커넥터를 사용하여 컴포넌트를 조합한다. 이는 커넥터 수의 증가를 피할 수 없어 컴포넌트 간의 의사소통이 비효율 적으로 이루어지는 문제가 발생한다. 본 논문에서는 계층적 메시지 전달 방식을 제안하고 이를 사용한 조합 방법을 통해 이상적인 컴포넌트 생명 주기를 따르는 새로운 소프트웨어 컴포넌트 모델인 트리 아키텍처 컴포넌트 모델을 제안한다. 아울러 제안된 모델의 적용을 통해서 컴포넌트 사이를 중재하는 객체를 사용하지 않고도 쉽게 재사용가능할 뿐만 아니라 의사소통도 효율적으로 가능함을 발견할 수 있었다.

1. 서론

CBSE는 현재 많은 연구를 통해 컴포넌트 모델에 관한 상당한 성과를 이루었다 [1, 2]. 그러나 이들은 독점적인 프레임웍과 컴포넌트 명세서를 요구하고 이를 사용하기 위한 학습에 많은 시간이 필요하다. 그리고 이들의 컴포넌트는 재사용하기 쉽지 않거나 조합 메커니즘이 잘 정의되지 않아 시스템 적으로 적용하기가 어렵다 [3]. 이와 같은 EJB, CORBA, COM, KobrA 등의 이전의 모델들은 RPC 메소드와 이벤트 위임과 같은 직접 메시지 전달 방법을 사용하였으며 이는 순환 호출이 발생하고 제어와 계산이 분리되지 않는 등의 몇 가지 심각한 문제를 가지고 있어 컴포넌트의 재사용을 어렵게 하였다 [4].

이런 문제점을 해결하고 컴포넌트의 재사용성을 확장하기 위해 간접 메시지 전달을 사용한 모델이 연구되었으며 이들은 컴포넌트 사이의 연결을 위해 커넥터 또는 추상화의 다른 단계에서 커넥터로 해석될 수 있는 조합 연산자를 기반으로 조합하였다 [4, 5, 6]. 그러나 이와 같은 연구도 시장이 원하는 컴포넌트 재사용의 목표를 충분히 달성하지는 못하였다.

최근에는 소프트웨어 컴포넌트의 재사용을 위한 여러 연구들에서 공통적으로 받아들이는 이상적인 컴포넌트의 생명주기가 제안되었고 이를 실현하는 컴포넌트 모델 또한 제안되었다 [3, 4, 7, 8]. 이 모델은 이전의 컴포넌트 모델의 문제를 해결하기 위해 간접 메시지 전달 방식을 사용한 Exogenous 커넥터를 제안하여 이상적인 컴포넌트 모델에서 추구하는 설계 단계와 배포 단계에서 컴포넌트의 조합을 가능하게 하였다. 하지만 이 방법은 커넥터의 수의 증가를 가져와 컴포넌트들 사이의 의사소통이 비효율적으로 이루어지는 문제점을 가지고 있다 [4, 8].

본 논문에서는 이상적인 컴포넌트 생명주기를 실현하고 기존의 문제점을 해결하기 위해 계층적 메시지 전달을 제안한다. 이는 메시지 전달을 효율적으로 하면서 컴포넌트의 연결과 제어를 계산과 구분하여 수행하는 것이 가능하다. 이를 바탕으로 컴포넌트의 구현, 조합 방법을 설명하여 이상적 컴포넌트 생명 주기를 실현하는 새로운 컴포넌트 모델을 제안한다. 이를 트리 아키텍처 컴포넌트 모델이라고 부른다.

본 논문의 구성은 2장에서 관련 연구로 이상적인 컴포넌트 생명주기 그리고 컴포넌트 모델에 관해 살펴보고, 3장에서는 트리 아키텍처 컴포넌트 모델을 설명한다. 4장에서는 3장에서 제안한 모델을 사용한 구현의 예를 든다. 마지막으로 5장에서 결론을 기술한다.

2. 관련연구

이상적인 컴포넌트 생명주기를 따르는 소프트웨어 컴포넌트 모델을 제안하기 위해서 이상적 컴포넌트 생명주기와 이를 구현한 기존의 모델에 관한 장단점을 알아본다. 그리고 기존의 모델을 개선하기 위해 필요한 이산 수학과 자료구조에서의 Tree 구조를 알아볼 것이다.

2.1 이상적인 컴포넌트 생명주기

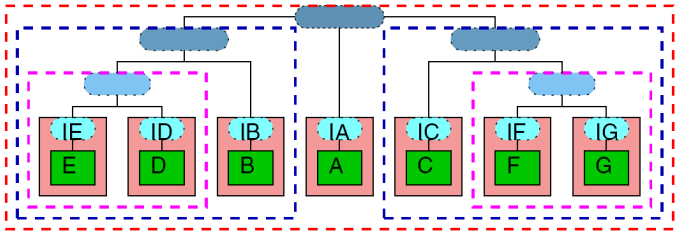
기존의 컴포넌트 생명주기는 설계 단계, 배포 단계, 실행 단계 세 가지 단계로 구성 된다. 여기에서 이상적인 컴포넌트의 조합은 설계 단계와 배포 단계 둘 다에서 가능할 것이다. 조합을 하는 것은 컴포넌트의 재사용을 의미한다. 그러나 기존의 컴포넌트 모델들은 두 단계 중 한쪽에 지나치게 치우쳐 있다 [8]. 이는 컴포넌트의 재사용 가능성을 축소시킨다. 그러므로 이상적인 컴포넌트 생명주기는 [3, 7, 8] 두 단계 모두에서 조합에 의해 개발 될 수 있도록 하여 설계의 유연성과 컴포넌트의 재사용을 최대화 할 것이다.

2.2 이상적인 컴포넌트 생명주기를 따르는 기존의 모델

최근에는 이상적인 컴포넌트 생명주기를 따르는 모델이 제안되었다 [8]. 이것은 기존의 커넥터들의 문제점을 해결하기 위해 Exogenous 커넥터를 사용해 조합을 한다. 이 모델은 다음의 특징을 가진다 [4].

- 제어와 처리가 혼합되지 않는다.
- 외부의 컴포넌트로부터 메소드 호출이 시작된다.

위의 특성을 통해 제어는 커넥터에서만 발생하고 계산은 컴포넌트에서만 발생한다. 그리고 커넥터를 중심으로 컴포넌트들이 계층적으로 연결되고 모든 컴포넌트들은 커넥터를 통해서 제어가 이루어진다. 이를 설명하는 조합 계층 구조는 (그림 1)과 같다.



(그림 1) Exogenous 연결

Exogenous 커넥터를 사용한 계층적 구조는 본 논문에 많은 영감을 주었다. 그러나 이 모델은 Exogenous 커넥터의 잠재적인 큰 증가를 피할 수 없어 컴포넌트간의 메시지 전달 의사소통이 매우 비효율적으로 이루어지는 문제가 있다. 본 논문에서는 위의 문제점을 개선하고 이상적인 컴포넌트 생명주기를 따르는 새로운 모델을 제안한다.

2.3 Tree

이산수학에서 트리는 어떠한 순환도 포함하지 않는 비순환 그래프로 정의한다 [9]. 그리고 자료구조의 관점에서 트리는 노드들을 간선으로 연결한 계층형 자료구조를 의미한다 [10]. 본 논문에서는 트리의 노드를 컴포넌트로 간선을 두 컴포넌트사이가 연결 된 것으로 대응시킨다.

3. 트리 아키텍처 컴포넌트 모델

소프트웨어 컴포넌트 모델의 핵심은 조합이다 [3, 7]. 이는 컴포넌트의 재사용을 의미한다. 특히 이상적 컴포넌트 생명주기는 설계 단계와 배포 단계 둘 다에서 조합을 지원해야한다. 본 논문의 모델에서는 컴포넌트의 조합을 위해 계층적 메시지 전달 방법을 사용한다. 이는 기존의 이상적 컴포넌트 생명주기를 따르는 모델의 비효율적인 의사소통문제를 개선한 방법이다.

3.1 계층적 메시지 전달 방법

기존의 EJB, CORBA, COM, KobrA 등의 모델에서 직접 메시지 전달을 사용하였다. 그러나 이는 순환적 호출 같은 문제를 방지하는 방법이 없어 개발자의 능력에 의존해야하고 컴포넌트들 내부에 제어와 계산이 혼합되어 재사용을 어렵게 한다 [4].

본 논문에서는 컴포넌트 사이의 메시지 전달을 위한 아키텍처를 제안하고 이를 따라 메시지를 서로 주고받는 프로토콜을 제안하여 컴포넌트 사이의 제어를 수행한다. 이는 트리 구조를 이루는 파이프라인을 따라 모든 컴포넌트가 동일한 방법의 제어를 따르게 됨을 의미한다. 다음은 제어를 위한 연결과 프로토콜을 정의하는 방법을 설명한다.

- 트리 아키텍처를 사용한 컴포넌트 연결 - 트리는 비순환 구조를 가지는 그래프이다 [9]. 트리의 노드를 컴포넌트로 간주하고 연결하여 트리를 따라 컴포넌트가 의사소통을 하면 순환호출의 문제를 해결한다. 트리를 이루는 방법으로 컴포지트 패턴[11]을 사용한다.
- 메시지 전달을 위한 프로토콜 인터페이스 구현 - 트리 구조를 따르는 의사소통을 위해 컴포넌트는 동일한 메시지 전달 방법을 사용해야한다. 이를 위해 데코레이터 패턴 [11]을 사용하여 구현하고 이는 컴포넌트가 의사소통을 위한 동일한 제어를 가지도록 한다.

제어를 위해 필요한 프로토콜 인터페이스를 정의하기 위해 본 논문에서는 트리의 연결 관점과 구성하는 노드의 관점에서 인터페이스를 정의한다. 트리 아키텍처는 위에서 아래로의 연결과 아래에서 위로의 연결이 존재한다. 그리고 트리를 구성하는 노드들은 3가지 종류가 있다. 이들은 루트, 잎, 그리고 중간의 노드들이다. 루트는 위에서 아래로의 연결만 필요하고, 잎은 아래에서 위로의 연결만 필요하다. 그리고 중간의 노드는 둘 다 필요할 것이다. 따라서 제어를 위한 연결 인터페이스는 상위에서 하위로의 인터페이스와 하위에서 상위로의 연결만 필요하다. 그리고 컴포넌트를 노드로 대응하고 인터페이스들과 함께 적용하면 컴포넌트들도 3가지 종류로 구성될 수 있다. 이때 중간의 노드는 가지 컴포넌트로 부른다.

트리 아키텍처의 제어를 위한 연결에는 (표 1)에서와 같이 각 컴포넌트 별로 인터페이스를 구현해야 한다. 루트 컴포넌트는 최상위의 컴포넌트로 하위의 컴포넌트만 연결된다. 따라서 하위를 위한 인터페이스만 필요하다. 잎 컴포넌트는 최하위 컴포넌트로 상위의 컴포넌트만 연결되기 때문에 상위를 위한 인터페이스만 필요하다. 가지 컴포넌트는 상위와 하위에 모두 연결되므로 상위와 하위를 위한 인터페이스가 모두 필요하다.

컴포넌트	구현 인터페이스
루트 (Root)	InterfaceForHigher
잎 (Leaf)	InterfaceForLower
가지 (Branch)	InterfaceForHigher, InterfaceForLower

(표 1) 컴포넌트 종류 별 구현 인터페이스

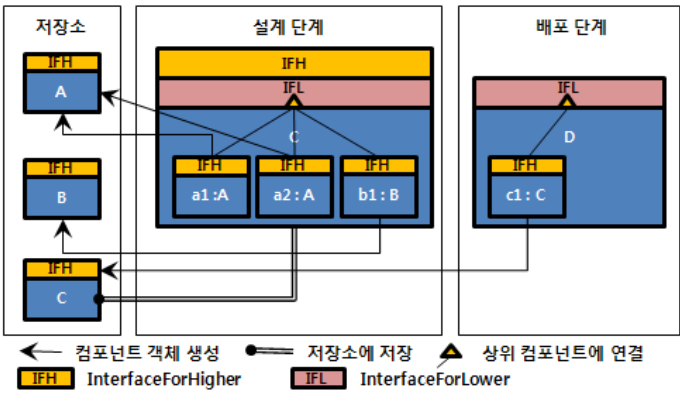
3.2 컴포넌트 구현의 두 가지 방법

본 논문이 제안하는 모델에서 컴포넌트를 만드는 방법은 2가지 경우가 존재한다 : (1) 새로운 컴포넌트를 만드는 경우, (2) 트리 아키텍처 컴포넌트 모델을 사용하지 않는 이미 자신의 인터페이스가 존재하는 컴포넌트를 사용하는 경우

(1)는 컴포넌트의 서비스 계산 로직을 구현하고 각 컴포넌트의 종류에 따라 필요한 인터페이스를 구현하면 된다. (2)는 기존의 컴포넌트가 서비스를 제공하고 있지만 인터페이스가 일치하지 않는다. 하지만 본 논문에서의 컴포넌트 정의는 객체지향에서 발전한 바이너리로 제공되는 컴포넌트이므로 트리 아키텍처 모델에 맞는 컴포넌트로 변경하기 위해 어댑터 패턴 [11]을 사용한다. 어댑터가 되는 컴포넌트는 이미 존재하는 대상 컴포넌트를 어댑티로 상속받고 컴포넌트의 종류에 따라 앞에서 제안한 인터페이스를 구현한다.

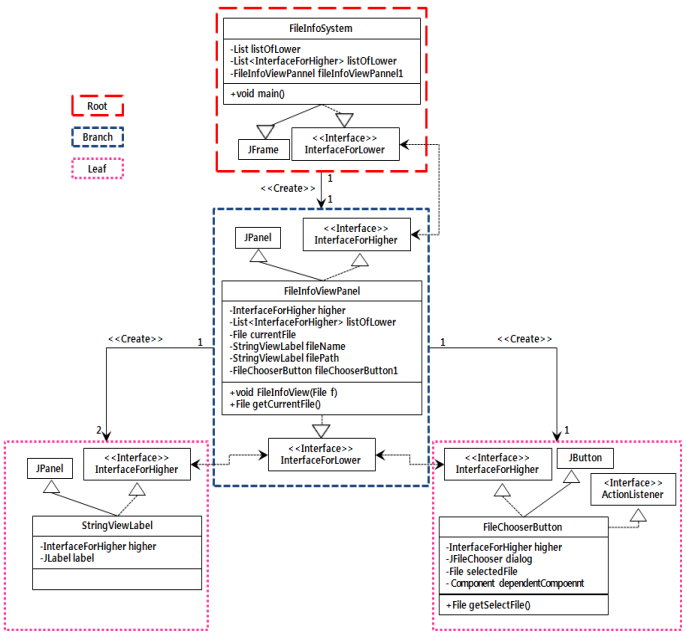
3.3 컴포넌트 조합 방법

본 논문에서 트리 아키텍처 컴포넌트 모델은 이상적인 컴포넌트 생명주기를 따르기 때문에 조합은 설계 단계와 배포 단계에서 이루어진다. (그림 2)의 C 컴포넌트와 같이 설계 단계의 조합은 저장소의 컴포넌트들을 재사용하여 생성하고 컴포넌트 사이를 연결을 한다. 이와 같이 설계 단계의 조합으로 새로운 서비스를 제공하는 계산 로직을 구현한 컴포넌트를 저장소에 다시 저장한다. 만약 사용할 컴포넌트가 저장소에 없다면 잎 컴포넌트로 처음부터 구현하고 이것을 저장소에 저장한다. 배포 단계의 조합은 (그림 2)의 D 컴포넌트와 같이 저장소의 컴포넌트를 재사용하여 생성과 연결을 하고 구현 후 루트 컴포넌트가 되어 시스템으로 실행이 가능한 상태가 된다.



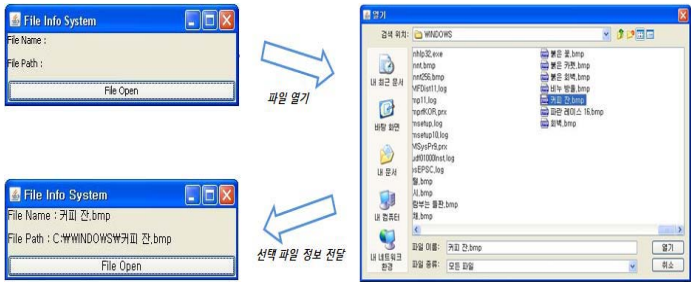
(그림 2) 트리 아키텍처 컴포넌트 모델의 조합

4. 구현



(그림 3) FileInfoSystem의 컴포넌트 아키텍처

트리 아키텍처 컴포넌트 모델을 적용하여 파일 정보를 확인하는 간단한 시스템을 구현한다. 구현 시스템의 조합 아키텍처를 그리면 (그림 3)과 같다. 이는 (그림 2)에서 묘사한 것을 구현 하는 시스템으로 D는 루트 컴포넌트와 대응되고 C는 가지 컴포넌트와 대응한다. 그리고 a1, a2는 (그림 3)의 왼쪽의 잎 컴포넌트와 대응하고 b1는 (그림 3)의 오른쪽 잎 컴포넌트와 대응한다.



(그림 4) File Info System의 실행 결과

본 구현에서 컴포넌트간의 제어가 이루어지는 과정을 살펴보면 FileChooserButton 컴포넌트에서 외부의 사용자의 이벤트를 받는다. 이때 사용자에의해 선택된 파일이 프로토콜 인터페이스로 정의된 메소드를 사용해 상위 컴포넌트에게 전달된다. 상위 컴포넌트는 전달 받은 메시지(선택된 파일)를 이를 필요로 하는 하위 컴포넌트에게 직접 전달하여 각 컴포넌트의 기능을 제어한다. (그림 4)은 시스템의 실행과 결과이다.

5. 결론

본 논문은 트리 아키텍처 컴포넌트 모델을 제안하였다. 이는 이전의 연구에서 공통적으로 받아들이는 컴포넌트의 특성을 바탕으로 만들어진 이상적 컴포넌트 생명주기를 따르는 소프트웨어 컴포넌트 모델이다. 이 모델은 기존 모델 [8]의 문제점인 메시지 전달의 비효율성 문제를 개선하기위해 컴포넌트 사이를 중재하는 객체를 사용하지 않고 인터페이스를 사용하여 쉽게 재사용 가능하고 의사소통도 효율적인 컴포넌트의 합성방법을 제안한다. 또한 랩핑을 사용한 컴포넌트의 연결과 제어 방법은 인터페이스가 일치하지 않는 컴포넌트의 재사용 또한 가능하여 레거시 컴포넌트의 재사용성도 크게 확대되었다. 이것은 컴포넌트의 재사용을 극대화하기 위한 새로운 조합 방법이다.

추가적으로 기대되는 효과로는 계층적으로 조합된 구조를 통해 시스템을 익숙한 트리 구조로 파악할 수 있고 이는 시스템적으로 컴포넌트를 조립 시 기본 구조로 사용할 수 있을 것이다.

앞으로의 연구로는 본 모델을 적용한 프레임웍의 개발과 이를 지원하는 도구의 개발을 통해 이상적인 컴포넌트 생명주기의 실현을 가능하게 하는 연구가 필요 할 것이다.

[참고문헌]

[1] Szyperski, C., Gruntz, D., Murer, S., "Component Software: Beyond Object-Oriented Programming Second Edtion," ADDISON-WESLEY, 2002

[2] Xiaoqin, X., Peng, X., Juanzi, L., Kehong, W., "A Component Model for Designing Dynamic GUI: Parallel and Distributed Computing, Applications and Technologies," In Proc. of PDCAT'2003. the Fourth International Conference, pp. 136-140, 2003

[3] Lau, K.-K., Wang, Z., "Software Component Models," IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 33, No. 10, pp. 709-724, 2007

[4] Lau, K.-K., Velasco Elizondo, P., Wang, Z., "Exogenous Connectors for Software Components. In," Heineman, G.T., Crnković, I., Schmidt, H.W., Stafford, J.A., Szyperski, C.A., Wallnau, K. (eds.) In Proc. of CBSE 2005. LNCS, vol. 3489, pp. 90-106. ,2005

[5] 임윤선, 김명, 정승남, 정안모, "컴포넌트 재사용을 지원하는 컴포넌트 모델 및 프레임워크," 한국정보과학회, v.34, no.12, pp.1011-1020, 2007

[6] Washizaki, H., Fukazawa, Y., "A Model-View Separation Structure for GUI Application Components," Information Technology: Coding and Computing, In Proc. of ITCC 2005. International Conference Volume 2, pp.359 - 364 Vol. 2, 2005

[7] Lau, K.-K., Wang, Z., "A Taxonomy of Software Component Models," In Proc. of 31st Euromicro Conf. Software Eng. And Advanced Applications (SEAA'05), pp.88-95, 2005

[8] Lau, K.-K., Ling, L., Velasco Elizondo, P., "Towards Composing Software Components in Both Design and Deployment Phases," In Proc. of CBSE 2007, LNCS 4608, pp. 274-282, 2007

[9] 황대훈, "이산수학," 생능출판사, 1993

[10] 이석호, "자료 구조와 JAVA," 정익사, 2003

[11] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, "Design Patterns," Addison-Wesley, 1995