아키텍쳐 기반의 컴포넌트 조립 시스템 설계 및 지원도구의 개발

이승연*, 권오천*, 신규상*
*한국전자통신연구원 컴퓨터소프트웨어연구소
e-mail : {coral, ockwan, gsshin}@etri.re.kr

Design and Tool of Component Assembly based on Architecture

Seung-Yun Lee*, Oh-Choen Kwon*, Gue-Sang Shin*
*ETRI Computer & Software Laboratory

요 약

복잡한 응용 프로그램을 빠르게 개발하고 이의 유지보수가 용이하도록 하기 위해 재사용 가능한 컴포넌트 기반의 소프트웨어 개발(CBD: Component-Based Development)개념이 확산되고 있고, 이와 관련되어 컴포넌트 재사용성을 높이기 위한 EJB, COM, CCM 등과 같은 컴포넌트 모델과 이의 개발 및 이미 개발된 컴포넌트를 재사용하고자 하는 연구가 다양하게 진행되고 있다. 컴포넌트를 기반으로 시스템을 효과적으로 구성하기 위해서는 컴포넌트들이 서로 정확하게 결합하여 작동할 수 있는 아키텍쳐를 기반으로 컴포넌트의 조립 작업이 이루어져야 한다. 잘 정의된 아키텍쳐가 존재하면, 시스템이 상위수준에서 어떻게 구성되어 있고, 어떠한 기능을 수행하는지의 방향을 제시해 주므로 조립하는 시스템이 제공하는 서비스를 추상화하여 표현할 수 있다. 본 논문은 제 삼자에 의해 개발된 컴포넌트들의 조립을 통한 시스템 개발을 지원하기 위하여 아키텍쳐 기반의 시스템 설계 방법을 제안하고 이를 지원하는 도구인 COBALT(Component-Based Application DeveLopment Tool) 조립도구의 아키텍쳐 모델러를 소개한다.

1. 서론

컴포넌트 기반 개발(CBD: Component-Based Development) 개념의 등장으로 소프트웨어 재사용을 통한 복잡한 시스템 개발을 위하여, 기존 컴포넌트들 간의 상호관계를 식별하고 어떻게 재사용할 것인지에 대한 연구와 함께 EJB, COM, CCM 등과 같은 컴포넌트 모델에 대한 연구가 활발히 진행되고 있다[1]. 제 삼자에 의해 개발된 컴포넌트들을 조립하여 시스템을 개발할 때에는, 코드의 수정이나 재컴파일 없이 다양한 플랫폼에 원하는 컴포넌트들을 전개하여 사용될 수 있어야 한다. 그러나, 재사용하고자 하는 식별된 컴포넌트간의 상호작용이 가능하기 위해서는 컴포넌트 구현레벨에서의 직접적인 메소드 호출 로직이 필요하다[2].

이러한 문제를 해결하기 위해서는 컴포넌트들이 서로 정확하게 결합하여 작동할 수 있는 기반구조가 마련되어야 하고, 이를 바탕으로 플러그 앤 플레이 방식의 유연한 재구성 및 조립이 지원되어야 한다. 제공된 컴포넌트들의 조립을 시스템의 설계단계에서 아키텍쳐를 기반으로 컴포넌트의 조립구조를 나타낼 수 있어야 하며, 설계된 아키텍쳐를 바탕으로 시스템 개발에 사용될 구현 컴포넌트와 연결될 수 있어야 한다[3]. 잘 정의된 아키텍쳐가 존재하면, 시스템이 상위수준에서 어떻게 구성되어 있고, 어떠한 기능을 수행하는지의 방향을 제시해 주므로 조립하는 시스템이 제공하는 서비스를 추상화하여 표현할 수 있다. 또한, 아키텍쳐 분석과정을 통하여 구성된 시스템의 구조적 특징 및 행위적 특징을 찾아내고 이들의 의미적 정확성 및 완전성, 상호 일관성등을 검사할 수 있으며 아키텍쳐 정제과정을 통하여 구성된 모델의 요소들에 재사용 가능한 컴포넌트들을 정확하게 매핑시켜 주고 이들간의 관계를 지속적으로 유지해 줄 수 있다. 따라서 아키텍쳐를 기반으로 컴포넌트들을 조립하면 아키텍

쳐 모델링 및 분석 과정을 통하여 컴포넌트들이 정확하게 결합하여 작동될 수 있다.

본 논문은 제 삼자에 의해 개발된 컴포넌트들의 조립을 통한 시스템 개발을 지원하기 위하여 아키텍쳐 기반의 시스템 디자인 방법을 제안하고 이를 지원하는 도구인 COBALT(Component-Based Application DeveLopment Tool) 조립도구의 아키텍쳐 모델러를 소개한다. 2 장에서는 관련연구에 대해 기술하고, 아키텍쳐 기반의 컴포넌트 조립 설계 방법을 3 장에서 제안한다. 4 장에서는 제안하는 설계방법을 지원하는 COBALT 아키텍쳐 모델러를 소개하고 5 장에서 결론을 맺는다.

2. 관련연구

2.1 컴포넌트 조립

컴포넌트 조립(Component Assembly)은 컴포넌트 기반 소프트웨어 개발(Component Based Development: CBD)의 궁극적 목표로서, 소프트웨어를 구성하는 단위 컴포넌트들은 조립을 통하여 다른 컴포넌트들과 상호 작용한다. 컴포넌트간의 상호작용을 지원하기 위해서 컴포넌트는 인터페이스(Interface)를 기반으로 결합된다. 인터페이스는 정보 은닉(Information Hiding) 원리에 따라 컴포넌트를 자유롭게 대체하고 독립적으로 사용할 수 있도록 지원한다. 그러나, EJB 나 COM+와 같은 컴포넌트 모델에서의 인터페이스는 컴포넌트의 모듈 기능에 대한 표현은 제공하나, 컴포넌트 기능들간의 내부 흐름이나 입출력 흐름이 제공되지 않고, 컴포넌트간의 상호작용이 메소드 호출방식에 의해 수행되므로, ' Plug-and-Play' 방식의 손쉬운 조립을 행하는 데에도 많은 제약이 있다[4]. 이를 극복하기 위하여 아키텍쳐 레벨에서 정의하는 추상화된 인터페이스를 제공해야 하며, 컴포넌트간의 메소드 호출방식(Method Invocation)이 아닌 메시지 전달방식(Message Passing)을 통하여 컴포넌트 조립을 쉽게 지원할 수 있어야 한다.

2.2 아키텍쳐 스타일(Architectural Style)

아키텍쳐 스타일은 자주 사용되는 관용적인 소프트웨어 시스템의 구조 패턴을 의미하며 컴포넌트와 커넥터들로 모델링된다. 아키텍쳐 스타일은 구조적인 구성 패턴을 공유하는 시스템들의 집단을 정의하며, 아키텍쳐의 설계에 사용되는 컴포넌트 타입들, 커넥터 타입들, 그리고 이들의 상호 연결 패턴에 대한 규칙들과 제약 조건들을 정의한다[5].

아키텍쳐 스타일 중 C2 아키텍쳐 스타일은 미국의 UCI 대학에서 연구 개발중인 스타일로서 메시지 기반의 계층 아키텍쳐를 나타낸다[6]. 모든 컴포넌트와 커넥터는 top 과 bottom 을 가지고 있어 C2 컴포넌트들 사이의 통신은 커넥터를 통하여 이뤄지며 이 때에는 top, bottom 포트를 통하여 일정한 규칙을 따라 상호작용할 수 있다. 그림 1]은 C2 아키텍쳐 스타일의 개념을 나타낸 그림이다[7].

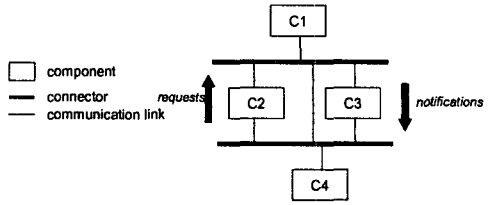

그림 1 C2 아키텍쳐 스타일

C2 컴포넌트의 구조는 EJB 컴포넌트를 독립적으로 만들 수 있는 wrapper 를 생성하여 메시지 패싱을 통한 컴포넌트간의 상호작용을 지원하는데 응용될 수 있다. 본 논문에서 소개하는 COBALT 아키텍쳐 모델러는 C2 아키텍쳐 스타일에 기반하여 시스템을 설계한다.

3. 아키텍쳐 기반의 컴포넌트 조립 설계 방법

본 장에서는 재사용 가능한 컴포넌트를 조립하여 새로운 어플리케이션 시스템을 아키텍쳐를 기반으로 설계하는 방법을 제안한다. 그림 2]은 조립 시스템을 개발하는 전체 프로세스와 이 중 시스템을 설계하는 방법을 세분화하여 나타낸 것이다.

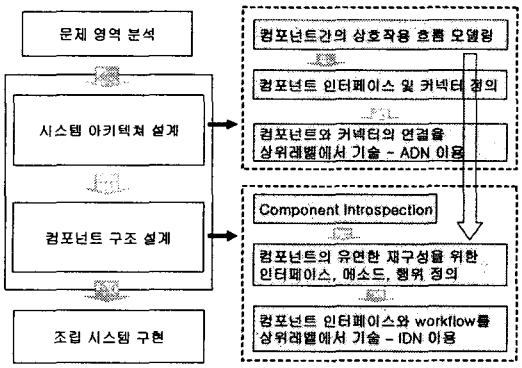

그림 2 조립 시스템 개발 프로세스 및 설계방법

조립 시스템의 개발 프로세스는 다음과 같다.

- 영역 분석 및 모델링: 제공된 컴포넌트들을 이용하여 어플리케이션 시스템을 설계하기 위해서는 도메인에 대한 이해와 사용자의 요구사항에 대한 분석이 선행되어야 하는데, 영역 분석 및 모델링 과정을 통하여 시스템 조립에 이용될 컴포넌트들을 식별한다.
- 어플리케이션 시스템 아키텍쳐 설계: 식별된 컴포넌트들은 조립 어플리케이션 시스템의 구조와 컴포넌트간의 상호관계를 나타내는 시스템 아키텍쳐 설계와 식별된 컴포넌트들과 시스템 구조를 연계시키기 위한 컴포넌트 구조 설계의 두 단계로 나뉘어 설계된다.
- 조립 시스템 구현: 설계된 아키텍쳐는 아키텍

처 기술 언어(ADL: Architecture Description Language)로 기술되며 이를 이용하여 실제 조립 어플리케이션 시스템을 구현하는데 필요한 wrapper와 접속코드를 생성한다.

어플리케이션 시스템 아키텍처는 시스템 아키텍처 설계와 컴포넌트 구조 설계의 두 단계로 나눌 수 있다.

시스템 아키텍처는 문제 영역 분석 단계를 거쳐 나온 결과를 이용하여 설계한다. 이 때 가장 먼저 수행되는 작업이 컴포넌트간의 행위 모델링이다. 식별된 컴포넌트를 이용하여 시스템의 요구사항을 만족하도록 이들간의 상호작용을 순차도로 표현한다. 컴포넌트 각각이 수행하는 기능을 인터페이스를 통하여 접근하도록 설계되어야 컴포넌트들의 유연한 재구성을 지원할 수 있는데, 정의된 인터페이스는 커넥터에 의해 관리된다. 설계된 컴포넌트간의 행위와 인터페이스는 C2 스타일에서 지원하는 ADN(Architecture Description Notation)으로 정형화한다[8].

컴포넌트 구조의 설계는 컴포넌트의 기능과 속성을 포트 기반의 메시지 전달 방식으로 추상화시켜 표현함으로써, 기존 컴포넌트의 코드수정 없이 컴포넌트간의 상호관계를 명확하게 표현하고 이들을 조립할 수 있는 기반을 마련해 준다. 제 삼자에 의해 개발된 컴포넌트의 기능과 속성은 Reflection 과정을 통하여 분석되며 분석된 결과와 시스템 아키텍처 설계 시 모델링되었던 컴포넌트 행위를 이용하여 시스템 영역에서 컴포넌트가 반드시 제공해야 하는 기능들을 정의하고 이를 메시지 전달 방식의 인터페이스로 표현한다. 정의된 컴포넌트의 구조는 C2 스타일의 IDN(Interface Definition Notation)으로 기술한다.

그림 3]은 EJB 컴포넌트의 조립을 지원하기 위해 정의한 컴포넌트 및 아키텍처의 Semantic 모델로서 시스템 아키텍처 설계시 기술되는 ADN 과 컴포넌트 구조 설계시 기술되는 IDN 으로 표현된다.

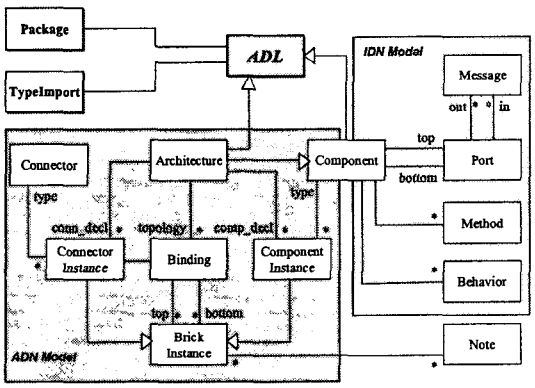

그림 3 컴포넌트 조립을 지원하는 아키텍처의 Semantic 모델

4. 아키텍처 기반의 시스템 설계를 지원하는 COBALT 아키텍처 모델러

본 장에서는 3 장에서 제안한 아키텍처 기반의 컴포넌트 조립 설계방법을 지원하는 COBALT 아키텍처 모델러를 소개한다. 문제영역 분석을 통하여 추출된 컴포넌트들을 C2 스타일에 맞게 설계하기 위하여 본 도구는 다음과 같은 기능을 수행한다.

- 시스템 아키텍처 편집 기능: C2 스타일을 기반으로 컴포넌트 조립을 지원하는 아키텍처는 다이어그램과 텍스트를 이용하여 기술할 수 있다. 시스템 아키텍처 설계는 다이어그램 편집기에서 컴포넌트를 끌어다 놓아 커넥터를 이용하여 조립 아키텍처를 생성하거나, 텍스트 기반의 편집기에서 C2 스타일을 따르는 아키텍처 기술언어를 이용하여 조립 아키텍처를 기술할 수 있다. 동일한 어플리케이션 구축을 위해 작성된 컴포넌트 조립 아키텍처의 다이어그램 기술내용과 텍스트 기술내용은 상호 일치해야 한다.

- 컴포넌트 구조 편집 기능: 각각의 컴포넌트에 대해서 컴포넌트 구조를 설계하기 위한 편집기능은 컴포넌트의 인터페이스, 메소드, 행위를 정의할 수 있어야 한다. 이를 컴포넌트 명세 작성 단계라고 하는데 제 삼자에 의해 개발된 컴포넌트 정보를 분석할 수 있는 기능이 제공되어야 한다. 컴포넌트 구조의 설계는 시스템 아키텍처 설계와 마찬가지로 다이어그램 상에서 작성하는 방법과 텍스트 편집기를 이용하여 작성할 수 있다.

- 모델 검사 기능: 다이어그램이나 텍스트로 작성된 아키텍처 모델은 구문검사와 의미검사를 통하여 문법 및 의미검사를 수행할 수 있다. 구문검사는 아키텍처 모델이 문법에 맞게 작성되었는지를 말하며 의미검사에는 타입검사와 형세(topology) 검사가 포함된다. 타입 검사란 아키텍처 모델이 아키텍처 언어의 타입규칙에 맞게 작성되었는지를 말하며 형세 검사란 아키텍처 스타일에서 규정하는 규칙에 따라 아키텍처 구성요소가 연결되고 구조화되었는지를 말한다.

위의 요구사항을 만족하기 위해 포함되어야 하는 아키텍처 기반 조립지원도구의 기본 구성은 크게 시스템 아키텍처 편집기, 컴포넌트 편집기, 모델검사기로 나눌 수 있다. 편집기는 아키텍처 다이어그램 편집기와 컴포넌트 속성/명세 편집기, ADL 편집기, 그리고 컴포넌트 정보 분석기가 있다. 모델검사기는 편집기에서 작성된 아키텍처의 구문 및 타입검사를 담당하며 이를 통하여 올바른 아키텍처가 설계되었는지 검증한다. 그림 5]는 COBALT 아키텍처 모델러의 전체 구성도 및 산출물과의 관계를 나타낸 것이다.

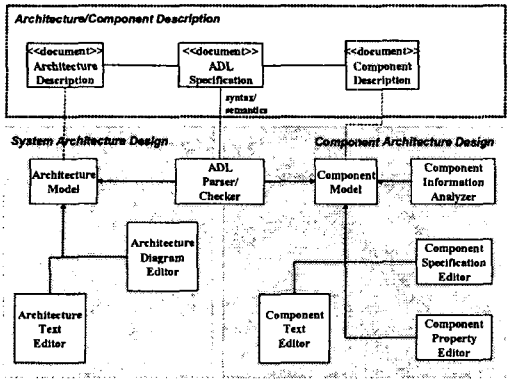

그림 4 COBALT 아키텍쳐 모델러의 구조 및 단위 모듈과 산출물간의 관계

아키텍쳐 다이어그램 편집기나 아키텍쳐 텍스트 편집기를 이용하여 작성된 아키텍쳐 모델은 아키텍쳐 명세파일로 저장되며, 이는 ADL 파서/검사기를 통하여 검사된다. 아키텍쳐를 구성하는 각각의 컴포넌트는 컴포넌트 속성/명세편집기를 통하여 컴포넌트 모델을 생성하며 이때 컴포넌트 정보 분석기에서 분석된 컴포넌트 정보를 이용하게 된다. 생성된 컴포넌트 모델은 컴포넌트 명세파일로 저장된다. 이렇게 구성된 아키텍쳐 모델러는 COBALT 조립도구의 코드생성기능과 연동하여 조립 시스템 개발의 전체 프로세스를 지원한다.

그림 5]는 COBALT 아키텍쳐 모델러에서 시스템 아키텍쳐 설계와 컴포넌트 구조 설계를 지원하는 과정을 나타낸 것이다.

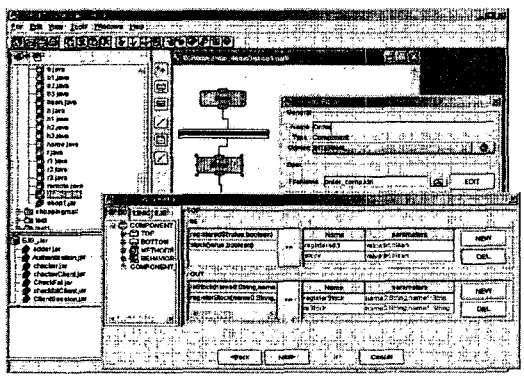

그림 5 시스템 아키텍쳐 및 컴포넌트 구조 설계 과정

5. 결론 및 향후연구

본 논문은 컴포넌트 조립에 의한 응용 소프트웨어 시스템의 개발을 효과적으로 지원하기 위해 아키텍쳐를 기반으로 컴포넌트들이 서로 정확하게 결합하여 작동할 수 있도록 지원하는 설계 방법을 제안하고 지원 환경을 구현한 COBALT 아키텍쳐 모델러를 기술하였다.

아키텍쳐를 기반으로 기존에 개발된 컴포넌트 모델을 연결하여 조립하면 아키텍쳐 모델링, 분석, 정제의 과정을 통하여 컴포넌트들이 정확하게 결합하여 작동될 수 있다. 본 논문은 이러한 장점을 반영하기 위하여, 아키텍쳐를 기반으로 시스템의 조립을 유도하였으며, 컴포넌트의 유연한 조립을 위하여 컴포넌트의 아키텍쳐를 메시지 전달 방식으로 표현하고 이를 지원하는 도구를 개발하였다. 현재, 컴포넌트 조립을 통한 시스템 개발의 전 단계를 지원하는 도구를 개발하였으며 이의 사례적용을 통하여 컴포넌트 조립시 아키텍쳐를 기반으로 설계하고 구현하면 컴포넌트의 대체 및 추가등의 시스템 유지 보수도 손쉽게 처리할 수 있음을 확인하였다. 향 후, 컴포넌트 조립을 지원하기 위한 아키텍쳐 기술 언어의 확장이 필요하며 이를 지원하는 도구의 확장이 필요하다.

참고문헌

[1] Andersson, J., Johnson, P., Architectural integration styles for large-scale enterprise software systems, Proceedings of 5th International Enterprise Distributed Object Computing Conference, pp 224 -236, 2001.

[2] Felix B., Len B., Charles B., Santiago C.D., Fred L., John R., Robert S. and Kurt W., Technical Concepts of Component-Based Software Engineering, Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 2000.

[3] Moreira, R.S., Blair, G.S., Carrapatoso, E, A reflective component-based and architecture aware framework to manage architecture composition, Proceedings of 3rd International Symposium on Distributed Objects and Applications(DOA 2001), pp 187 -196, 2001.

[4] Felix B., Len B., Charles B., Santiago C.D., Fred L., John R., Robert S. and Kurt W., Technical Concepts of Component-Based Software Engineering, Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 2000.

[5] M. Shaw, "Patterns for Software Architectures", Proceedings of PLoP '94 Conference, PLoPD, Addison Wesley, pages 453-462, 1995.

[6] Nenad Medvidovic, Davis S. Rosenblum, and Richard N. Taylor, A Language and Environment for Architecture-Base Software Development, In the Proceedings of the 21st International Conference on Software Engineering (ICSE 21), Los Angeles, CA, pages 44-53, May, 1999.

[7] Nenad Medvidovic, Peyman Oreizy, and Richard N. Taylor, "Reuse of off-the-shelf components in C2-style architectures", Proceedings of the Symposium on Software Reusability (SSR'97), pp 190-198, Boston, MA, May, 1997.

[8] 신동익, 노성환, 최재각, 권태웅, 이승연, 권오현, 신규상, C2 스타일의 아키텍쳐 기술을 지원하는 ADL 지원도구의 개발, 정보처리학회 논문지 D 제 8-D권, 제 6호, 2001년 12월.