아키텍쳐 모델링을 위한 UML 2.0 프로파일

노성환* 김경래* 전태웅* 승현우**
고려대학교 전산학과* 서울여자대학교 정보통신공학부**
{shroh, blueage98, jeon}@selab.korea.ac.kr* hwseung@swu.ac.kr**

UML2.0 Profile for an Architecture Modeling

Sunghwan Roh* Kyungrae Kim* Taewoong Jeon*, Hyonwoo Seung**
Dept. of Computer Science, Korea University*
School of Information & Communication, Seoul Women's University**

요 약

UML을 기반으로 소프트웨어 아키텍쳐를 모델링하기 위한 많은 연구가 진행되어 왔다. 특히 UML을 확장(extension)하여 UML이 지원하지 않는 핵심적인 아키텍쳐 개념을 명시적으로 표현하기 위한 노력이 진행되어 왔다. 그러나 기존의 대부분의 연구들은 UML1.x을 기반으로 한다. 이에 비해 공식 발표될 예정인 UML2.0은 아키텍쳐 모델링에 관한 개념을 보다 많이 포함하고 있다. 하지만 UML2.0에도 명시적인 표현이 어려운 아키텍쳐의 핵심 개념들이 여전히 존재한다. 또한 UML2.0은 아키텍쳐 기술에 불필요하거나 관련이 적은 모델링 요소들도 많이 포함하고 있다. 본 논문에서는 UML2.0에서 표현된 아키텍쳐 모델링 요소를 살펴보고, 아키텍쳐를 표현하기 위해서 UML2.0을 어떻게 아키텍쳐 기술 언어에 대한 프로파일로 확장하고 정의할것인지를 고려한다.

1. 서 론

대부분의 ADL들은 각 소프트웨어 아키텍쳐의 특정한 면에 촛점을 맞추어 개발되어 왔다. 대표적인 ADL들은 Adage[1], Aesop[2], Darwin[3], Rapide[4], SADL[5], Wright[6] 등이다. ACME[7]와 같은 2세대 ADL은 기존 ADL들의 공통적인 개념을 뽑아내어 통합하고 있다. 그러나 ACME 사용자는 각 ADL의 특정 표기형식(notation)을 익혀야 하기 때문에 ACME는 아키텍쳐 명세를 위한 언어로 정착되지 못하였다. 반면 UML은 범용 모델링 언어로서 소프트웨어 개발의 전 과정에 일관된 표기형식과 폭넓은 지원도구들을 제공하고 있으며, 소프트웨어 개발을 위한 사실상의 표준 언어이다. UML과 같은 표준 언어로서 아키텍쳐를 명세함으로서 아키텍쳐는 이해하기 쉽게되고, 개발 과정 동안 일관성을 유지하게되며, 기존 개발 도구들의 지원을 받을 수 있게 된다. 또한 UML로 표현된 아키텍쳐 모델은 설계 모델과 구현 모델로 쉽게 변경될 수 있다. 이러한 이유로 UML을 이용하여 아키텍쳐를 모델링하기 위한 연구들이 많이 진행되어 왔으며, 본 논문에서는 아키텍쳐 모델링을 위해 UML을 확장하여 프로파일로 정의하는 방법을 연구한다.

2. 관련 연구

소프트웨어 아키텍쳐를 UML로 표현하기 위한 표현하기 위한 기존의 연구들은 크게 두가지 방법으로 나뉠 수 있다. 첫번째 방법은 기존의 UML 표기형식을 변경 없이 아키텍쳐 개념에 적용하여 사용하는 방법이다. 두번째 방법은 기존 UML의 어휘와 의미를 아키텍쳐의 개념에 맞도록 확장하여 사용하는 방법이다. 두번째 방법은 다시 heavyweight 방법과 lightweight 방법으로 나뉜다. Heavyweight 방법에서는 UML 메타모델을 직접 수정하여 새로운 구조물을 추가하거나 기존의 의미를 변경한다. Lightweight 방법에서는 UML 메타모델은 변경하지 않으면서 UML의 확장 메커니즘을 사용해 새로운 구조물을 정의한다.

본 연구는 한국과학재단 목적기초연구(R01-2002-000-00044-0)와 한국전자통신연구원 위탁연구의 지원으로 수행되었음.

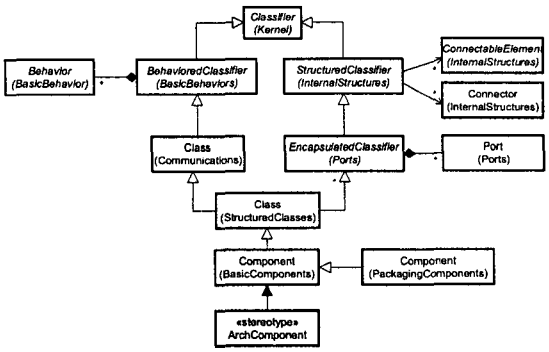
[8] 등의 논문에서 heavyweight 방법에 기반하여 UML을 확장한 아키텍쳐 기술 언어를 제안하고 있지만 기존 툴의 지원을 받을 수 없는 문제가 있다. [9,10,11,12,13] 등 다수의 논문에서 lightweight 방법에 기반한 UML 언어 확장을 제안하고 있으나 이들은 모두 UML 1.x 버전을 기반으로하고 있다. UML 2.0[14]은 Component, Connector, Port 등 아키텍쳐 기술을 위한 구조물들을 이전 버전의 구조물들을 확장하거나 새롭게 추가하여 정의하고 있다.

[15]에서는 이러한 UML 2.0을 이용하여 아키텍쳐 기술을 위한 프로파일을 정의하고 있다. [15]는 ACME의 개념에 직접적으로 대응하도록 프로파일을 정의하고 있다. 그러나 ACME에서는 포트를 일종의 인터페이스로 취급하고 있는 반면, UML 2.0에서는 포트와 인터페이스가 별개의 개념으로 구분되어 있다. 이 때문에 [15]에서 정의된 아키텍쳐 인터페이스의 개념 및 표기가 부자연스럽다. 또한 커넥터는 컴포넌트 메타클래스의 스테레오 타입으로 정의되어 컴포넌트와 동일한 표기를 사용하며 이로 인해서 아키텍쳐 모델의 가시성이 떨어진다. 또한 확장된 UML2.0 구조물들과 메타모델과의 의미적인 일관성이 명확하지 않다.

3. UML 2.0 기반의 아키텍쳐 모델링 언어의 정의

3.1. 컴포넌트(Component)

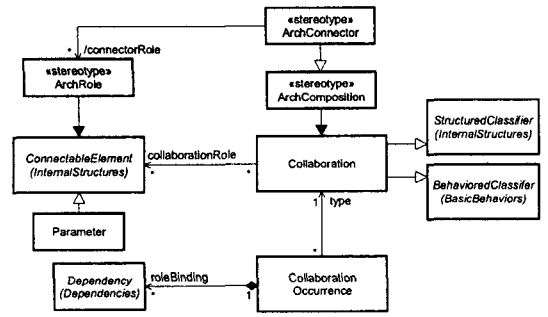
이전 버전의 UML에서의 컴포넌트는 논리적인 실행 단위로서의 의미와 내부 구조 및 상호작용 지점을 정의할 수 없었던 문제가 있었다. 그러나 UML 2.0에서는 이러한 개념을 모두 포괄할 수 있도록 컴포넌트의 정의가 확장되었다. 따라서 UML 2.0 Component는 아키텍쳐 상의 독립적으로 실행가능한 합성 단위로서의 컴포넌트를 정의하는데 충분한 모델링 요소로 볼 수 있다. 그러나 UML2.0 Component가 갖는 Feature 중에 아키텍쳐 상의 컴포넌트가 갖는 특성과 직접적으로 관계되지 않는 것들은 제약을 가할 필요가 있다. [그림 1]의 ArchComponent는 Component 메타클래스의 스테레오타입으로 정의되며 이러한 제약 사항이 추가되었다.



[그림 1] Generic ADL의 컴포넌트

3.2. 합성 패턴(Composition Pattern)

UML2.0의 Collaboration은 Structured Classifier이면서 Behaviored Classifier로서 협동하는 개체들의 집합의 구조적 측면과 행위적 측면을 함께 표현할 수 있으며, 컴포넌트 인스턴스들의 연결관계로서 표현되는 컴포넌트 간 합성 패턴이나 아키텍처의 구성을 나타내는데 이용될 수 있다. 따라서 [그림 2]에서와 같이 UML 2.0 Collaboration을 베이스 메타클래스로 하여 Generic-ADL에서 합성 패턴을 나타내는 모델링 요소인 ArchComposition을 정의한다. ArchComposition에서 합성 패턴에 참여하는 role들의 의미를 제약하여 ArchRole로서 정의한다. ArchRole은 roleBinding에 의해 ArchPort에만 대응되도록 제약한다.

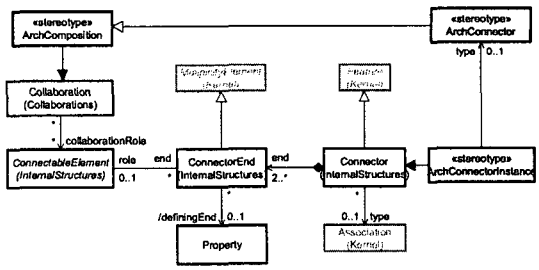


[그림 2] Generic ADL의 합성 패턴

3.3. 커넥터(Connector)

UML 2.0의 Connector는 타입이 아닌 인스턴스 수준의 개념이다. 이에 따라 본 논문에서는 ArchConnector를 UML 2.0 Collaboration의 스테레오타입으로 정의하여 타입 수준의 커넥터를 표현하였다. 그리고 인스턴스 수준의 커넥터인 ArchConnectorInstance를 UML 2.0 Connector의 스테레오타입으로 정의하여 인스턴스 수준의 커넥터를 표현하였다. 이렇게 정의된 ArchConnectorInstance는 자신에 연결되는 ConnectableElement의 유형이 Port로 한정되며 ArchConnetor를 타입으로 가질 수 있다.

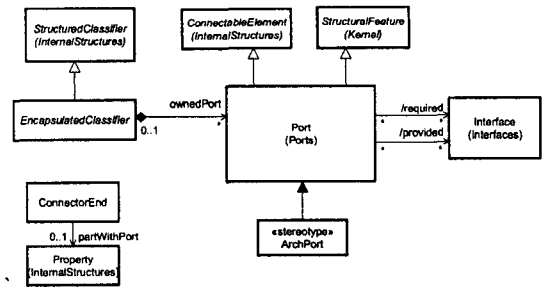
커넥터가 단순한 의미와 구조를 갖는 경우(예를 들어 delegation 용도로 사용되었거나 두개 포트 사이에 required, provided 인터페이스의 단순 연결로 사용되는 경우 등)에 ArchConnectorInstance는 UML 2.0의 Connector와 동일한 의미를 갖으며 타입으로 ArchConnector를 갖지 않는다. 그러나 커넥터가 복잡한 행위적 의미와 구조를 갖는 경우 (예를 들어 복잡한 프로토콜에 의거한 상호작용, 상호작용에 참여하는 컴포넌트의 역할 명시, 상호작용 변환 기능, 또는 빈번하게 나타나는 연결 패턴의 의미하는 경우 등)에 UML 2.0의 Connector 만으로는 표현이 어렵기 때문에 ArchConnector는 ArchComposition를 특화하여 정의한다.



[그림 3] Generic ADL의 커넥터

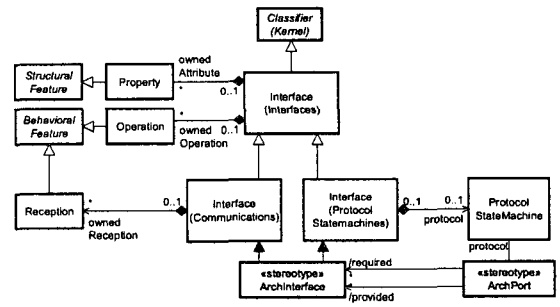
3.4. 포트(Port)

컴포넌트의 대체 가능성(Substitutability)과 연결 가능성(Connectability)의 측면에서 중요한 상호작용 지점을 UML 2.0에서는 Port로 새롭게 정의하였다. UML2.0의 Port는 컴포넌트가 외부로 제공하는 인터페이스 명세 뿐만 아니라 외부로부터 필요로 하는 인터페이스 명세를 나타내기 위해서 provided interface 이외에 required interface를 갖는다. 이러한 UML2.0 Port의 스테레오타입으로 [그림 4]의 ArchPort를 정의한다. 한편 Connector가 컴포넌트 간 연결이 아니라 컴포넌트의 내부 구조를 표현하는데 사용되는 경우 Connector의 connectorEnd에 연결되는 것은 해당 컴포넌트의 Property로서의 connectableElement의 타입에 따르는 Port가 된다.



[그림 4] Generic ADL의 포트

3.5. 인터페이스(Interface)



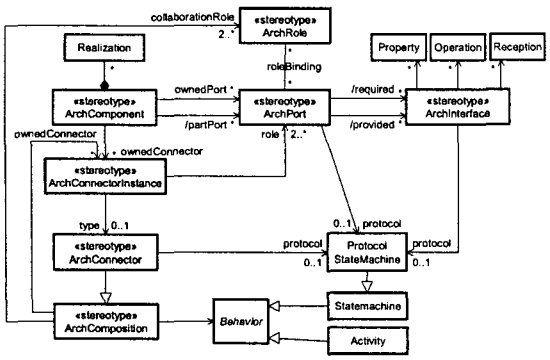
[그림 5] Generic ADL의 인터페이스

UML 2.0의 Port가 갖는 Interface는 메소드 호출과 같은 상호작용 뿐만 아니라 비동기적인 시그널을 처리할 수 있도록 Reception을 지원한다. 또한 인터페이스가 갖는 행위적 의미에 대한 명세가 가

능하도록 ProtocolStatemachine을 가질 수 있다. [그림 5]의 ArchPort가 갖는 ArchInterface는 이러한 UML 2.0 Interface을 확장하여 정의된다. ArchPort가 갖는 인터페이스는 ArchInterface만 가능하도록 의미를 한정한다. ArchPort의 행위적 의미는 이러한 ArchInterface들의 ProtocolStatemachine의 합성으로 표현될 수 있다.

3.6. 정의된 Generic ADL 의 메타모델

[그림 6]은 앞에서 기술한 Generic ADL의 메타모델이다. [그림 8]에서 흰색 바탕으로 표시된 모델 요소들은 UML 2.0의 메타클래스이고 어두운 바탕색은 UML 2.0 메타클래스의 스테레오타입으로 정의되어 Generic ADL에 새롭게 추가된 모델 요소들이다. ArchComponent는 자신이 소유한 외부 Port(owndedPort)로서 ArchPort를 갖고 ArchComponent의 내부 구조를 구성하는 part들의 연결점으로서 partPort를 갖는다. 컴포넌트의 내부 구조는 포트들의 연결 관계로 정의된다. ArchPort는 자신이 속한 컴포넌트가 제공하는 행위적 특성(provided interface)과 필요로 하는 행위적 특성(required interface)을 나타내기 위해 ArchInteface를 참조한다. ArchInterface는 Property, Operation, Reception을 피처(feature)로 가짐으로써 동기적, 비동기적인 상호작용을 모두 표현할 수 있다. ArchComposition은 컴포넌트 간의 합성 패턴을 나타내며 UML2.0 Collaboration의 스테레오타입으로 정의된다. ArchRole은 Collaboration의 role에 참여하는 개체를 ArchPort로서 한정함으로 정의된다. ArchRole은 CollaborationOccurrence의 roleBinding 관계를 이용하여 ArchPort에 대응된다. 합성 관계를 나타내는 커넥터는 ArchConnectorInstance로 한정한다. ArchConnector는 Arch Composition의 서브 타입으로 정의되며 ArchConnectorInstance의 타입에 해당된다. ArchConnectorInstance는 자신의 인터페이스로서 ArchPort를 참조할 수 있는데, 이 ArchPort의 타입은 ArchConnectorInstance의 타입인 ArchConnector의 ArchRole에서 roleBinding을 통해 대응되는 ArchPort의 타입에 맞아야 한다 (conform to).



[그림 6] Generic ADL의 메타모델

4. 결론 및 향후 연구

본 논문에서는 UML 2.0에 기반한 아키텍처 모델링 언어의 설계 방법을 연구하였다. 본 논문에서 정의된 메타모델은 주로 아키텍처의 구조적인 면을 주로 다루고 있으며 아키텍처의 행위를 명세하는데에는 충분하지 않다. 그리고 아키텍처 모델로부터 구현 모델로의 정제과정을 지원하기 위한 연구가 필요하다. 정제 과정은 분할(Decomposition)과 서브타이핑(Subtyping)의 개념을 이용할 수 있다. 또한 소프트웨어 개발 환경이 복잡화 됨에 따라 다양한 컴포넌트 간 연결이 나타나고 있는데, 커넥터가 이러한 다양한 연결 방식들을 충분히 표현할 수 있는지에 대한 검증이 필요하다. Generic ADL의 메타모델에 대한 연구 이후에는 이것을 확장하여 영역에 종속적인 ADL을 개발하고 이로써 실제 다양한 영역의 아키텍처를 설계하고, 분석할 수 있도록 연구할 예정이다. 또한 추상화 수준에 따른 계층적인 ADL의 정의는 MDA 기반의 소프트웨어 개발에 이용될 수 있다. 각 계층 ADL 메타모델 간 매핑 관계를 정의함으로써 개념적 아키텍쳐 모델로부터 구현 수준의 설계 모델로의 연결 관계가 명확해질 수 있으며, 따라서 빠르고 신뢰성 있는 소프트웨어를 적은 비용으로 개발하는데 기여할수있다.

참고 문헌

[1] Coglianese, L. and Szymanski, R. DSSA-ADAGE: An Environment for Architecture-based Avionics Development. In Proceedings of AGARD'93, May 1993.

[2] Garlan, D., Allen, R., and Ockerbloom, J. Exploiting style in architectural design environments. In Proceedings of SIGSOFT'94: The Second ACM SIGSOFT Symposium on the Foundations of Software Engineering, pages 179{185. ACM Press, December 1994.

[3] Magee, J., Dulay, N., Eisenbach, S., and Kramer, J. Specifying distributed software architectures. In Proceedings of the Fifth European Software Engineering Conference, ESEC'95, September 1995.

[4] Luckham, D. C., Augustin, L. M., Kenney, J. J., Veera, J., Bryan, D., and Mann, W. Specification and analysis of system architecture using Rapide. IEEE Transactions on Software Engineering, Special Issue on Software Architecture, 21(4):336-355, April 1995.

[5] Mark Moriconi and R.A. Riemenschneider: Introduction to SADL 1.0 : A Language for Specifying Software Architecture Hierarchies,TR SRICSL -97-01, March 1997

[6] Allen, R. and Garlan, D. A formal basis for architectural connection. ACM Transactions on Software Engineering and Methodology, July 1997.

[7] David Garlan, Robert T. Monroe, and David Wile: Acme: Architectural Description of Component-Based Systems, Foundations of Component-Based Systems, Gary T. Leavens and Murali Sitaraman (eds), Cambridge University Press, 2000, pp. 47-68.

[8] Jorge Enrique, Perez Martinez: Heavyweight extensions to the UML metamodel to describe the C3 architectural style, ACM SIGSOFT Software Engineering Notes Volume 28 , Issue 3 (May 2003)

[9] Mohamed Mancona Kande, Valentin Crettaz, Alfred Strohmeier, Shane Sendall: Bridging the gap between IEEE 1471, an architecture description language, and UML. Software and System Modeling 1(2): 113-129 (2002)

[10] P. Selonen and J. Xu, "Validating UML Models Against Architectural Profiles", Proc. ESE/FSE'03, September 1-5, 2003, pp. 58-67

[11] Nenad Medvidovic, David S. Rosenblum, Jason E. Robbins, David F. Redmiles: Modeling software architectures in the Unified Modeling Language, ACM TOSEM Volume 11, Issue 1, 2 - 57

[12] A. Zarras, V. Issarny, C. Kloukinas, and V. K. Kguyen: Towards a Base UML Profile for Architecture Description, Proceedings of ICSE 2001 Workshop, 22-26, May 2001.

[13] Mohamed Mancona Kande and Alfred Strohmeier: Towards a UML Profile for Software Architecture Descriptions, UML'2000, LNCS, no. 1939, 2000, pp. 513-527.

[14] UML 2.0 Superstructure, 3rd Revision, OMG document ad/03-04-01, Object Management Group, 2003,

[15] Miguel Goulao, Fernando Brito e Abreu: Bridging the gap between Acme and UML 2.0 for CBD, Workshop at ESEC/FSE 2003 - September 1-2, 2003