VisDiS:분산 시스템 설계를 위한 시각적 아키텍처 기술 언어

정인복○ 김형호 배두환
한국과학기술원 전산학과

VisDiS : A Visual Architecture Description Language supporting Distributed System Design

In Bok Chung, Hyung Ho Kim, and Doo Hwan Bae
Dept. of Computer Science, KAIST

요 약

컴포넌트 기반 개발 방법이 확산됨에따라 소프트웨어 아키텍처는 핵심 기술중의 하나로서 각광받고 있다. 컴포넌트 기반의 아키텍처의 연구와 함께 정형적 아키텍처 기술을 지원하는 많은 아키텍처 기술 언어들이 제시되었다. 그러나, 메시지 전달 패러다임을 기술하고 분석할 필요가 있는 분산 시스템의 경우 기존의 아키텍처 기술 언어들은 이러한 기능의 지원이 부족하다. 본 논문에서는 메시지 전달 패러다임을 명확하게 기술할 수 있고, 이를 기반으로 적합성을 검사할 수 있으며, 일반 개발자들이 이용하기 편리한 시각적 아키텍처 기술 언어를 제시한다. 이를 통하여 분산 시스템을 설계하는 경우, 좀 더 정확한 분석과 정보를 제공할 수 있고, 시각적 언어로서 일반 개발자들이 좀 더 편리하게 아키텍처를 설계할 수 있게 된다.

1. 서론

컴포넌트 관련 기술이 성숙해 지고 크고 복잡한 소프트웨어를 원하는 시간 내에 시장에 출하하는 것이 중요해 짐에 따라서 컴포넌트 기반 개발 방법 (Component-Based Software Development:CBSD)는 더욱 중요성을 더해가고 있다. 컴포넌트 기반 개발 방법을 이루는 관련 기술들 중 하나인 소프트웨어 아키텍처(Software Architecture)는 시스템의 구조적 기술과 시스템을 구성하는 컴포넌트와 그 시스템을 구성하는 컴포넌트들 사이의 상호작용을 기술한 것이다[2]. 소프트웨어 아키텍처는 컴포넌트 기반 개발 방법에서의 컴포넌트 사이의 조합에 대한 기술 및 분석을 제공함으로써, 컴포넌트 기반 개발 방법의 핵심 기술 등 중 하나로 자리잡고 있다. 또한, 아키텍처를 정형적으로 기술하는 아키텍처 기술 언어에 대한 연구의 필요성이 제기 되면서 많은 아키텍처 기술 언어들이 제시되어 왔다[4].

그러나, 분산 시스템의 소프트웨어 아키텍처를 기술하고 분석할 경우, 메시지 전달 패러다임인 동기적/비동기적 메시지 전달을 구분하여 고려될 필요성을 충분히 반영하는 방법에 부족함이 있다. 분산 시스템을 이루는 각 컴포넌트에 메시지 전달 패러다임에 대한 정보를 기술하지 않는 경우, 두 컴포넌트가 서로 다른 메시지 전달 패러다임을 지니고 있다면 통신,상호작용중에 교착상태에 빠지거나, 오작동을 하게 되는 경우가 생길 수 있게 된다 또한, 정형 기법에 관한 지식이 부족한 일반 개발자의 경우 기술에 어려움을 가질 수 있다. 그러므로, 아키텍처 수준에서 분석을 통하여 이러한 문제점을 발견하고 설계이나 구현시에 지침을 제공하는 것을 가능하게 하기위해, 패러다임에 관한 기술을 하여주고 일반 개발자가 편리하게 아키텍처를 기술할 수 있는 방법이 필요하다.

따라서 본 논문에서는 이러한 부분을 보완하기 위해 동기적/비동기적 메시지 전달을 구분하여 기술할 수 있는 방법을 제시한다. 또한, 시각적 기술 언어의 제안과 아키텍처 기술 언어로 기술된 내용을 기반으로 아키텍처 수준에서의 분석을 지원하는 방법을 제시하도록 한다.

이 논문의 구성은 다음과 같다. 먼저 2장에서는 소프트웨어 아키텍처와 아키텍처 기술 언어에 대해 간략하게 설명하고 3장에서는 기존의 아키텍처 기술 언어 중 Wright, Rapide등과 같은 일부 특징적인 언어에 대해서 설명한다 4장에서는 시각적 설계 방법과 적합성(conformance) 검사 방법을 제시하고, 마지막으로 5장에서는 결론 및 향후 연구 방향을 제시한다.

2. 소프트웨어 아키텍처와 아키텍처 기술 언어

소프트웨어 아키텍처는 전체 시스템을 구성하는 컴포넌트와 그들 사이의 상호작용을 기술함으로써 컴포넌트 기반 개발에 있어서 전체 시스템 설계의 역할을 수행한다. 소프트웨어 아키텍처를 정형적으로 기술하고 효과적인 분석을 수행하기 위하여 정형적인 소프트웨어 기술 언어들이 제안되었다. 아키텍처 기술 언어는 다음과 같은 공통적인 부분을 갖는다.

- 컴포넌트에 대한 기술 : 다른 컴포넌트와의 통신하는 지점인 포트와 포트들 사이의 관계가 기술된다.
- 커넥터에 대한 기술 : 컴포넌트간의 상호작용의 행위, 통신 규약이 표현된다.

- 시스템의 구성에 대한 기술 : 기술된 컴포넌트와 커넥터를 실체화시키고 연결하여 전체 시스템의 구성이 기술된다.

3. 기존의 아키텍처 기술 언어들

현재까지 많은 아키텍처 기술 언어들이 범용 혹은 각 도메인에 알맞은 아키텍처를 기술하기 위해 제안되었다. 본 논문에서는 아키텍처 기술 언어들 중에서 특징적인 아키텍처 기술 언어에 대해 간략하게 살펴보도록 한다.

3.1 Wright

WRIGHT는 아키텍처의 커넥터를 정형화하고, 정형 시멘틱을 지니는 구성요소로 기술하는 것을 지원함으로써 컴포넌트간의 상호작용에 대한 기술을 강조한 언어이다[5]. 즉, 컴포넌트간의 상호작용을 CSP를 이용하여 기술하고, 이를 기반으로 기술함으로써 CSP에서 제공하는 교착상태(deadlock)에 대한 검증과 컴포넌트의 결합 가능성 분석(compatability checking)을 제공한다. 그러나, Wright는 CSP에 대한 익숙하지 못한 일반 개발자들이 이용하기 힘들고, 개발자를 위한 도구들의 지원이 충분하지 않다. 또한, 분산 시스템의 경우, 메시지 전달 패러다임의 구분을 명확하게 기술하지 못하며, 이에 대한 분석도 부족하다.

3.2 Darwin

Darwin은 분산 소프트웨어의 아키텍처를 기술하기 위해 정의된 선언적 결합 언어(declarative binding language)이다[3]. 특히, Darwin은 시스템의 정적인 구조(static structure)뿐만 아니라, 실행 중에 변경되는 동적인 구조(dynamic structure)에 대한 기술을 지원한다.

Darwin은 π-calculus를 기반으로 하여 시멘틱을 정의하며, 컴포넌트간의 상호 작용과 결합을 모델링(modeling)한다. Darwin은 아키텍처를 기술하고 분석하는 도구를 지원하며, 분산 환경에서의 시스템 설계나 관리등의 작업을 지원하는 환경이 개발되어 있다. 그러나, 분산 소프트웨어를 기술하는데 있어 필요한 메시지 전달 패러다임에 대한 기술과, 적합성 검사에대한 지원이 부족하다. 또한, 커넥터에 관한 기술이 단순한 결합관계만을 기술하여 컴포넌트들 간의 상호작용에 관한 분석의 지원에 부족함이 있다.

3.3 Rapide

Rapide는 시스템의 아키텍처를 프로토타이핑(prototyping)하기 위해 만들어진 이벤트 기반의 언어이다[1] 컴포넌트가 외부에 보여지는 행위에대한 기술과 컴포넌트의 인터페이스 모델링을 지원한다. 컴포넌트의 행위는 이벤트의 부분 정렬 집합(partially ordered set:poset)을 이용하여 정의된다. 이를 기반으로 시뮬레이션과 아키텍처의 행위의 분석을 지원한다. 또한, 기술된 시스템의 아키텍처를 이용하여 교착상태(deadlock)나 제약사항의 위배 등에 관한 분석이 가능하다. Rapide는 아키텍처를 기술하는 도구와 분석에 이용되는 도구들을 지원한다 그러나, 커넥터를 명백하게 분리하여 기술하지 않아 Wright와 같은 상호작용에 관한 분석의 지원이 부족하고, 메시지 전달 패러다임에 관한 충분한 분석이 이루어지지 않고 있다.

4. VisDiS:분산 시스템의 시각적 모델링 접근 방법

4.1 전체적 구조

본 논문에서 제시하는 ADL인 VisDiS는 시스템을 구조적 측면과 행위적 측면으로 기술한다. 시스템의 구조적 측면은 시스템을 이루는 컴포넌트들과 커넥터들이 어떻게 연결이 되었는지를 나타내게 된다 또한, 시스템의 행위적 측면은 시스템의 구조를 이루는 컴포넌트와 커넥터의 각 요소들이 시스템이 작동시에 어떠한 행위를 보이는 지를 나타내게 된다.

이렇게 기술된 정보를 이용하여 컴포넌트 간의 적합성 검사가 이루어질 수 있다.

4.2 아키텍처의 구조적인 측면의 기술

컴포넌트는 포트(port)들의 집합과 하나의 내부 연산 (computation)으로 이루어진다. 포트는 컴포넌트가 외부와의 통신에서 나타내는 행위를 기술한 것이고, 내부 연산은 이러한 포트들 사이의 관계를 이용하여 컴포넌트의 내부 행위를 기술한 것이다. 또한, 커넥터는 역할(role)들의 집합과 글루(glue)로 이루어진다. 글루는 컴포넌트들 사이의 통신 규약(protocol)을 기술한 것이고, 역할은 그러한 통신 규약에 참여 가능한 컴포넌트의 통신시에 나타내는 행위를 기술한 것이다. 포트와 롤을 대응시킴으로써 이렇게 정의된 컴포넌트와 커넥터가 연결된다.

시스템의 구조적 측면에 대한 예는 그림 1과 같다.

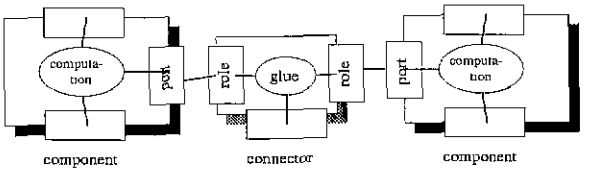

그림 1: 구조적 측면

4.3 아키텍처의 행위적인 측면의 기술

아키텍처 각 구성요소들 컴포넌트의 포트들과 내부연산, 그리고 커넥터의 롤들과 글루의 행위에 대한 기술은 확장된 FSM(Finite State Machine) 형태로 기술된다. 기술된 FSM을 이용하여 이벤트들의 나열 (event trace)을 생성할 수 있으며, 시스템의 행위는 발생한 이벤트의 나열의 집합으로 정의된다.

이벤트의 이름, 이벤트의 종류, 전달자(parameter)등과 같은 분석에 필요한 추가적인 정보들은 아크(arc)에 기술된다. 또한, 이벤트는 발생하는 이벤트(initiated event), 외부에서 발생한 것을 인지하는 이벤트(observed event), 그리고 실행을 통해 상태

를 변화시키는 ε 이벤트, 이렇게 3 종류로 구분된다. 그림 2는 커넥터의 글루와 롤의 행위를 보여주는 예이다

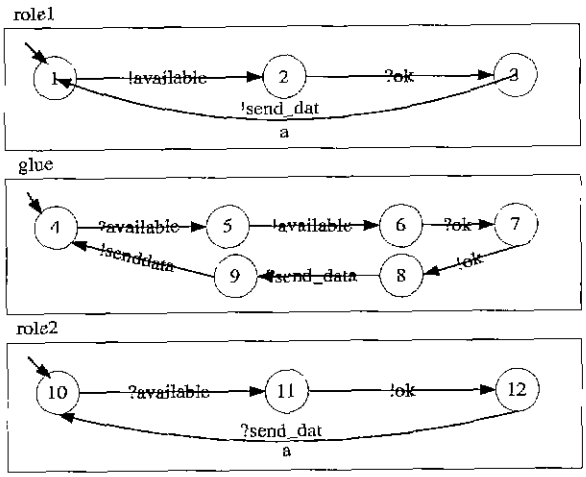

그림 2: 커넥터의 행위적 측면 기술의 예

그림 2에서의 아크 $\longrightarrow$는 비동기적 메시지 전달을 의미하는 것이다. 동기적 메시지 전달의 경우에는 아크 $\circ\longrightarrow$를 이용하여 기술함으로써 동기적 메시지 전달과 비동기적 메시지 전달을 구분한다.

4.4 적합성 검사

컴포넌트 간의 적합성(conformance)이란 시스템 구성 컴포넌트들의 기술 및 연결 상태가 시스템이 수행시 오작동하지 않고 주어진 요구 사항을 만족시키는 상태를 말한다.

시스템의 컴포넌트 간의 적합함을 보이기 위한 검사의 절차는 다음과 같다.

- 컴포넌트의 검사 : 컴포넌트내에 각 포트와 내부연산과의 관계에서 서로 일관성이 있는지를 검사한다.
- 커넥터의 검사 : 컴넥터의 프로토콜 상에서 교착상태가 발생하는지를 검사한다.
- 컴포넌트의 포트와 커넥터의 롤 간의 검사 · 포트의 행위가 롤의 행위에 포함이 되는지를 검사한다. 이것은 정적인 검사와 행위적인 적합성(behavioral conformance)검사로 이루어진다 즉,포트의 행위가 롤의 행위에 포함이 되는지의 여부는 행위적 적합성을 통해서 알 수 있게 된다.

교착상태 및 행위적 적합성에 관한 분석을 하기 위해 모델 검증과 같은 방법이 적용될 수 있다 특히, 본 논문에서 확장된 FSM을 의미론의 기반으로 하므로, SPIN과 같은 모델 검증 도구를 이용할 수 있을 것이다.

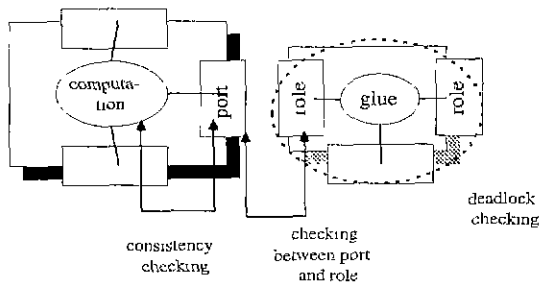

그림 3: 적합성 검사

5. 결론 및 향후 연구

본 논문에서는 메시지 전달 패러다임을 구분하여 기술하고 분석하는 방법을 제시하였다. 또한, 일반 개발자가 사용하기 편리하게 하기위해 시각적인 기술 언어인 VisDis를 제시하였다.

일반 개발자는 분산 시스템을 설계시에 메시지 전달 패러다임을 명확히 구분하여 기술함으로써 좀 더 정확한 분석을 할 수 있다. 또한, 그 결과로 얻은 정보는 세부 설계나 구현시에 이용될 수 있게 된다. 그러나, 일반 개발자들이 편리하게 검증을 하는 것을 돕기 위해, 기술된 것을 SPIN을 위한 입력 형태인 PROMELA로 변환시켜 주는 것이 필요하다. 향후 연구 과제로는 개발자가 기술한 정보를 이용하여 SPIN의 입력형태인 PROMELA로 변환시켜주는 방법에 대한 연구가 요구되어 진다. 또한, 시각적 아키텍처 기술 언어를 이용하여 기술할 수 있고, 기술된 것을 바탕으로 분석 도구인 SPIN의 입력형태인 PROMELA로 변환시켜주는 도구를 구현이 필요하다.

참고 문헌

[1] D. Luckham, et al,*Specification and analysis of system architecture using Rapide,IEEE TSE*, 21(4), Apr 1995

[2] D. Garlan and M. Shaw, *An Introduction to Software Architecture , Advances in Software Engineering and Knowledge Engineering*, Vol. 1, River Edge, NJ: World Scientific Publishing Company, 1993

[3] J. Magee, et al,*Specifying Distributed Software Architectures,Proc. of ESEC95* , Sep 1995

[4] N. Medvidovic, *A Classification and Comparison Framework for Software Architecture Description Languages*, University of California, Irvine *Technical Report* UCI-ICS-97-02, Feb 1996

[5] R. Allen and D. Garlan, *A Formal Basis for Architectural Connection,ACM TOSEM*, July 1997