내장형 시스템을 위한 아키텍처 모델링 언어 개발

문영주[01], 김문주[1], 김태효[2]
[1]한국과학기술원 전산학과
[2] (주) 포멀웍스
yjkmoon@kaist.ac.kr, moonzoo@cs.kaist.ac.kr, taihyo.kim@formalworks.com

Development of Architecture Description Language for Embedded Systems

Young-Joo Moon[01], Moonzoo Kim[1], TaiHyo Kim[2]
[1] CS Dept. Korea Advanced Institute of Science and Technology
2 Formal Works Inc.

요 약

기존의 아키텍처 모델링 언어는 내장형 시스템의 명세 및 검증의 중요성에도 불구하고, 그 언어들의 복잡하고 다양한 표기법으로 인해 전반적인 시스템 개발단계에서 사용이 저조한 실정이다. 이와 같은 문제를 해결하기 위해, 본 논문에서는 기존의 아키텍처 모델링 언어의 문법을 간략화하고, 개발단계에서 필요한 정보를 모델링단계에 명세할 수 있는 표기법을 제공하는 새로운 아키텍처 모델링 언어를 제안한다. 이와 더불어, 제안된 아키텍처 모델링 언어가 제공할 수 있는 다수의 명세 관점을 제안하고, 이를 바탕으로 내장형 시스템인 Anti-lock Braking System을 모델링한 결과를 예제로 보여준다.

1. 서 론

일상 생활에서 우리는 다양하고 많은 내장형 시스템을 마주하고 있다. 예를 들어, 간단한 스마트폰과 가전용품뿐만 아니라 자동차와 같은 안전성이 주요 쟁점이 되는 시스템의 내부에도 장착되어 사용된다. 따라서, 내장형 시스템의 행위의 정확성과 안전성을 검증하는 문제는 그 중요성이 점점 더 강조되고 있다.

이를 위해, 다양한 명세 및 검증 기법이 제시되었다 [1, 2]. 그러나, 시스템의 행위가 올바르게 명세 및 검증되더라도, 이를 바탕으로 구현한 결과물이 시스템의 요구사항을 반드시 만족시킨다는 것을 보장할 수 없다. 이는 시스템의 행위 단위요소들이 요소들이 실제 시스템의 구성요소에 전개(deployment)되는 과정에서 발생할 수 있는 예상치 못한 상황들을 고려하지 않기 때문에 발생한다. 따라서, 시스템 구성요소의 추상화, 시스템 구성의 스타일, 표준에 대한 표기법을 제시를 목표로, Acme [5], Wright [6], Architecture Analysis and Design Language (AADL) [7] 등과 같은 다양한 아키텍처 명세 언어(Architecture Description Language, ADL)들이 제안되었다. Acme는 다양한 ADL의 공통된 요소를 종합하여 범용 도메인에 사용을 목적으로 텍스트 및 그래픽 언어를 제공한다. 그러나, Acme는 특정 시스템의 특성에 대한 표현 제한이 있다는 단점이 있다. Wright는 언어의 범용적인 사용보다 시스템 컴포넌트간의 통신(interaction) 표현을 목적으로 텍스트 언어만 지원한다. 이는 실시간 내장형 시스템의 모델링에 적합하지 않을 뿐 아니라, 비기능적(non-functional) 속성 표현을 지원하지 않는다는 단점이 있다. AADL은 실시간 내장형 시스템 모델링을 위해 개발된 것으로 텍스트 언어와 그래픽 언어를 제공하며, 사용자 정의 가능한 비기능적 속성 표현을 지원한다. 그러나, AADL의 다양하고 복잡한 표기법으로 인해, AADL은 업계보다 학계에서 주로

사용되고 있으며, 업계에서는 주로 디자인 단계에서 각 이해관계자들의 의사소통의 수단으로만 사용되고 있다. 또한, AADL을 통한 모델링 결과가 실제 시스템 구현에 필요한 정보를 포함하지 않는다는 단점이 있다.

따라서, 본 논문에서는 실제 업계에서 전반적인 시스템 개발단계에서 사용될 수 있는 새로운 아키텍처 모델링 언어로, Simplified Architecture Description Language (SADL)를 제안하고자 한다. SADL은 복잡한 표기법은 간략화하고, 개발자가 내장형 시스템 구현에 필요한 정보를 모델링단계에서 명세할 수 있는 표기법을 제공한다. 이러한 SADL은 시스템을 명세하기 위해 소프트웨어적인 구성요소의 명세뿐만 아니라, 소프트웨어 구성요소들이 수행되는 물리적인 플랫폼인 하드웨어와 시스템이 상호작용하는 환경 요소에 대한 명세 표기법을 포함하고 있다. 또한, 제안된 SADL에서 제공할 수 있는 정보를 관점에 따라 독립적으로 살펴보기 위해 몇몇 명세 관점을 제안하고, 각 명세 관점 따라 제안된 SADL 표기법을 분류했다. 이를 바탕으로, Anti-lock Braking System (ABS)을 명세 관점에 맞추어 SADL로 모델링 한 결과를 보여준다. 따라서, 본 논문의 기여도는 다음과 같이 요약될 수 있다.

1) SADL 제안: 내장형 시스템의 소프트웨어와 하드웨어, 환경 요소 및 시스템 구현에 필요한 정보를 명세하는 간략한 표기법 제공
2) 명세 관점 제공: SADL이 지원하는 명세 관점 제공
3) SADL을 이용한 Anti-lock Braking System 모델링 예제

2. SADL (Simplified Architecture Description Language)

이 장에서는 내장형 시스템의 전반적인 개발단계에서

본 연구는 민군겸용기술사업(UM11014RD1)의 연구비 지원으로 수행하였음.

사용될 수 있는 새로운 아키텍처 모델링 언어 SADL을 정의한다. SADL은 엔티티(entity)와 연결 관계(connection)의 표기법으로 구성되어있으며, 이들은 각각 시스템을 구성하는 구성요소들과 그들 사이의 관계(relation)를 명세하기 위해 사용된다. SADL의 표기법은 기존의 아키텍처 모델링 언어에 비해, 구성요소 명세는 간략화하고, 구성요소간의 의존성 및 속성 정의 부분은 강화시켰다. 예를 들어, AADL과 비교했을 때, 원거리 서버에서 작동하는 프로그램과 같은 명세는 SADL에서 제외시켰다. 또한, 공유 데이터 사용(데이터 의존성)과 프로세스간의 의존성 등에 대한 명세를 첨가하여 시스템 구현에 좀더 근접한 명세가 가능하도록 했다.

2.1. 엔티티(Entity)

표 1 엔티티의 그래픽 표현형 및 의미

그룹	그래픽 표현형	의미
하드웨어	processor	소프트웨어 엔티티가 실행되는 플랫폼
	memory	시스템의 메모리
	connection	엔티티 사이의 데이터 흐름을 위해 사용되는 플랫폼
	device	시스템 외부의 물리적인 플랫폼으로, 주로 시스템에 정보를 전달해주는 센서와 외부 환경에 변화를 일으키는 구동장치(actuator)등을 표현
소프트웨어	unit	구조적 또는 기능적인 면에서 독립적으로 구현되는 단위
	process	unit 엔티티 내부 구성요소로 독립적으로 실행되는 단위
	unit group	여러 unit의 모임
	process group	여러 process의 모임
	data	소프트웨어 단위의 행위 객체
리소스	external resource	외부에서 주어지는 리소스

표 1에는 시스템 구성요소로써의 엔티티의 그래픽 표현형을 살펴볼 수 있다. 엔티티는 하드웨어, 소프트웨어, 리소스 3개의 그룹으로 나누어져 있고, 하드웨어는 processor, memory, device, connection으로, 소프트웨어는 unit, process, unit group, process group, data로, 리소스는 external resource로 이루어져 있다.

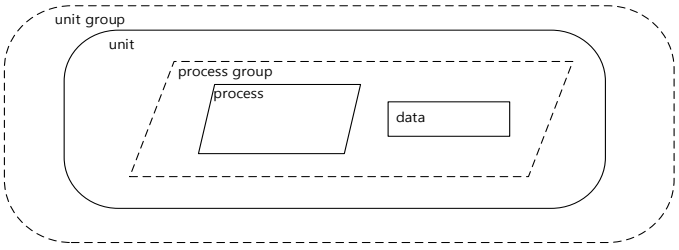

그림 1 소프트웨어 그룹 entity의 포함 관계

엔티티를 구성하는 3개의 그룹은 서로 독립적이다. 즉, 임의의 엔티티는 자신이 속한 그룹 이외의 다른 그룹에 속한 엔티티를 포함하거나 그 내부에 포함될 수 없다. 또한, 소프트웨어 그룹의 엔티티들은 일부 다른 소프트웨어 엔티티들을 내부에 포함한다. 이와 같은 포함 관계는 그림 1에서 살펴볼 수 있다. 각각 상위의 엔티티는 하위의 엔티티를 하나 이상 포함할 수 있다.

2.3. 연결 관계(Connection)

엔티티 연결 관계는 시스템 구성요소 사이의 관계를 명세하기 위한 것으로, 시스템 구성요소 사이의 정보흐름(data flow)과 의존성(dependency)을 표현한다. 이러한 연결 관계 R은 엔티티들의 집합인 Entity과 엔티티들간의 관계들의 집합인 Role={sends2, creates, uses, Ereads, Ewrites}[1]를 이용하여 다음과 같이 정의한다.

$$R \subseteq \text{Entity} \times \text{Role} \times \text{Entity}$$

엔티티 연결 관계 R은, 방향성을 지닌 화살표를 이용하여, 왼쪽과 오른쪽 엔티티를 각각 화살표의 소스(source)와 타겟(target)에 대응시키고, Role은 화살표의 레이블(label)로 사용하여 그래픽하게 표현한다. 표 2는 이러한 R의 그래픽 표현형과 각 연결 관계의 의미를 보여준다.

표 2 엔티티 사이의 연결 관계 명세

그래픽 표현형	의미
sends2 →	소스 엔티티에서 타겟 엔티티로 정보를 보냄
creates →	소스 엔티티가 타켓 엔티티를 생성함
uses →	소스 엔티티가 타겟 엔티티를 사용함
Ereads →	소스 엔티티가 타겟 엔티티에 배타적 읽기 수행
Ewrites →	소스 엔티티가 타겟 엔티티에 배타적 쓰기 수행

2.2. 엔티티 속성(Entity Attributes)

엔티티와 그들간의 연결 관계를 바탕으로 한 시스템 명세는 개발 단계에서 개발자가 자의적으로 결정해야 할 많은 부분을 여전히 남겨두고 있다. 예를 들어, connection의 비동기성 특성, process의 스케줄링 방침, 정보흐름의 타입과 같은 정보가 개발자에게 주어져야 한다. 이와 같은 정보들은, 모델링 단계에서 그래픽 표현형 외에 텍스트와 같은 다른 부가적 표기법으로 명세 된다. 표 3은 일반적인 엔티티의 속성을 보여준다. 페이지 제한으로 인해, 자세한 엔티티 속성의 정의 및 표현은 본 논문의 확장 버전에 첨부될 것이다.

[1] Ereads와 Ewrites는 각각 exclusively reads와 exclusively writes를 의미한다.

표 3 일반적인 엔티티 속성

종류	의미
name	엔티티 식별자
sub_entities	내부에 다른 엔티티를 포함하는 경우 표현.
protocol	1. memory의 경우, 접근 방침 표현 - 예) read/write, read only 2. processor의 경우, 스케줄일 방침 표현 - 예) FIFO, LIFO, round-robin

3. 명세 관점 (Description Viewpoint)

SADL은 시스템의 구성요소와 그들간의 관계를 표현할 수 있는 그래픽 표현형 및 구성요소들의 속성 명세를 통해, 시스템의 다양한 정보를 표현할 수 있다. 그러나, 하나의 명세서에 다양한 정보들이 한번에 표현이 되는 경우, 개발자에게 혼란만 가중시킬 수 있다. 이를 방지하기 위해, [3]에서는 의존성, 인터페이스, 결합, 상황(context), 구조, 상호작용 등 다양한 명세 관점을 제시하고, 각각의 명세 관점에 의해 여과된 시스템 정보를 개발자에게 제공하는 방법을 제안했다. 그러나, 각각의 명세 관점에 따른 다양한 시스템 정보 명세서를 사용하는 경우, 그들간의 표기법 일치성 문제와 각 명세서를 이해하기 위한 개발자들의 많은 노력이 요구된다는 점에서 실효성 문제가 제기된다 [4][2]. 따라서, 이 장에서는, 다양한 명세 관점 중 일부를, 즉, 결합, 의존성, 정보흐름 명세 관점, 채택하여 SADL의 표현 범위를 설명한다.

3.1 결합 명세

시스템을 구성하는 하드웨어와 소프트웨어의 구조적 연결 정보와 더불어 엔티티 사이의 사용/접근 정보 표현 및 소프트웨어 구성요소들의 계층 관계를 표현한다.

3.2. 의존성 명세

소프트웨어의 구성요소들과 리소스간의 사용/접근 정보, 소프트웨어 구성요소들 사이의 (배타적으로) 공유되는 정보에 의한 의존성 등을 표현한다.

3.3. 정보흐름 명세

소프트웨어와 하드웨어 사이의 정보흐름과 이와 관련된 정보의 타입 및 정보흐름을 야기시키는 process 등을 표현한다. 이를 위해, 소프트웨어와 하드웨어의 엔티티 및 이들 사이의 정보흐름 정보를 표현한다.

4. Anti-lock Braking System (ABS) 예제

ABS는 자동차와 운전자의 안전을 도모하기 위해 개발된 장치로, 자동차의 속도가 급작스럽게 변하는 경우 브레이크 압력을 조절하여 자동차가 안전하게 속도를 늦출 수 있도록 도와주는 역할을 한다. 이러한 ABS는 바퀴 속도 센서, 브레이크 페달 센서, 유압장치, 컨트롤 유닛으로 이루어져 있으며, 각 센서로부터의 정보는 CAN 버스를 통해 컨트롤 유닛으로 전달된다. 컨트롤 유닛은 일종의 컴퓨터로 빠른 계산을 통해 자동차 바퀴에 가해질 압력을 계산해서 그 결과값을 유압장치로 전달한다.

그림 2는 ABS를 결합 명세 관점으로 명세한 결과를 보여준다. 즉, ABS의 하드웨어와 소프트웨어 구성요소의 구조적 연결 정보와 구성요소들 사이의 사용/접근 정보 및 소프트웨어 구성요소들의 계층 관계를 보여주고 있다.

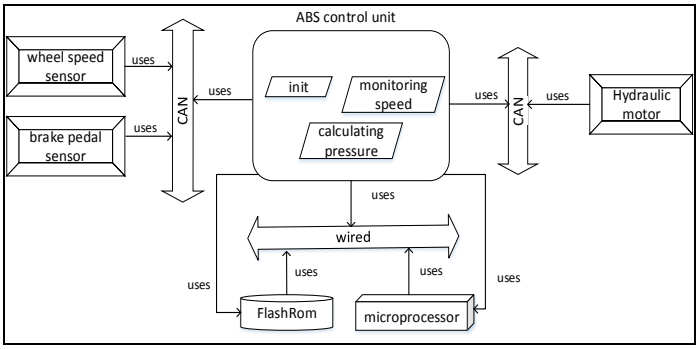

그림 2 결합 명세 관점에 따른 ABS 명세

5. 결론 및 향후 연구

본 논문에서는 내장형 시스템 구현에 적합한 아키텍처 모델링 언어로 SADL을 제안했다. 이는 기존의 아키텍처 모델링 언어를 간략화하고 동시에 개발자들의 의견을 바탕으로 한 표기법의 확대화를 통해 만들어졌다.

그러나, 아키텍처의 정보만으로는 내장형 시스템 행위의 정확성을 보장할 수 없다. 따라서 향후 연구로, SADL과 행위 모델링 언어와의 연계를 고려하고 있다. 특히, 내장형 시스템의 하이브리드 현상의 명세에 중점을 두고 하이브리드 명세 언어와의 연계를 살펴볼 예정이다. 이를 통해 시스템의 아키텍처 정보를 바탕으로 한 시스템 행위 모델링과 더불어 시스템 행위의 시뮬레이션 및 검증을 기대한다.

참고 문헌

[1] L. A. Cortés, P. Eles, and Z. Peng. "Verification of embedded systems using a petri net based representation." Proceedings of the 13th international symposium on System synthesis. IEEE Computer Society, 2000.

[2] R. Alur, T. A. Henzinger, and P.-H. Ho. "Automatic symbolic verification of embedded systems." Software Engineering, IEEE Transactions on 22(3) (1996): 181-201.

[3] IEEE Std 1016™-2009, IEEE Standard for Information Technology—Systems Design—Software Design Descriptions

[4] M. PETRE, UML in practice. In: Proceedings of the 2013 International Conference on Software Engineering. IEEE Press, 2013. p. 722-731

[5] D. Garlan, R. Monroe, and D. Wile. “Acme: an architecture description interchange language.” CASCON 1997

[6] R. Allen, D. Garlan. "A formal basis for architectural connection" ACM TOSEM 1997

[7] P. H. Feiler, D. P. Gluch, J. J. Hudak, and B. A. Lewis. ” Embedded system architecture analysis using SAE AADL” (No. CMU/SEI-2004-TN-005)

[2] [4]에서는 UML을 대상으로, 다양한 명세 관점의 실효성에 대해 언급하고 있다.