컴포넌트 정형명세를 위한 Z 스키마의 확장

이재희*, 장종표**, 김병기*
*전남대학교 전산학과
**한국정보통신기술협회

Extension of Z Schema for Component Formal Specification

Jae-Hee Lee*, Jong-pyo Jang**, Byoung-Gi Kim*
*Dept. of Computer Science, Chonnam National University
**TTA(Telecommumications Technology Association)
E-mail : jhlee@csblue.chonnam.ac.kr, jpjang@tta.or.kr, bgkim@chonnam.chonnam.ac.kr

요 약

컴포넌트를 개발하는데 있어서 컴포넌트 명세의 정확성과 명세의 검증을 통하여 에러를 찾아낸다는 것은 컴포넌트의 전체 품질에 매우 중요한 의미를 갖는다. 그러나, 기존의 컴포넌트 명세는 구문적인 측면은 잘 정의하고 있지만, 의미적인 측면은 자연어를 사용하여 모순과 모호성이 흔히 발생한다. 컴포넌트 명세에 있어서 정형적 문법을 사용할 경우 이러한 모호성을 제거함으로써 명세 오류들을 매우 효과적으로 줄여준다. 본 논문에서는 컴포넌트의 품질을 높일 수 있도록 분석력과 논리성이 검증된 정형 명세 언어 Z의 스키마 확장을 이용하여 컴포넌트를 명세하므로써 컴포넌트 구현 및 사용상의 오류를 분석할 수 있는 방법을 제안한다.

1. 서론

소프트웨어 위기를 해결하기 위한 대응책의 하나로 컴포넌트 기반 소프트웨어공학(Component Based Software Engineering)이 등장하였다. 관련 기술들 중 하나로써 컴포넌트기반 소프트웨어 개발 프로세서들이 제안되고 있다. 재사용을 목적으로 하는 컴포넌트 기반 소프트웨어 개발 프로세스에서 컴포넌트의 품질은 매우 중요하다. 이러한 컴포넌트를 개발하는데 있어서 컴포넌트 명세의 정확성과 명세의 검증을 통하여 에러를 찾아낸다는 것은 컴포넌트의 전체 품질에 매우 중요한 의미를 갖는다. 그러나, 기존의 컴포넌트 기반 소프트웨어 개발 프로세스에서는 컴포넌트 명세의 구문 적인 측면은 잘 정의하고 있지만, 의미적인 측면은 자연어를 사용하여 사용하기 편리한 반면 해석 시 모호성이 흔히 발생한다.[8] 의미적인 측면에서의 컴포넌트 명세는 올바른 컴포넌트 선택을 위해 필수적으로 선행되어야 하는 사항이며 부적절한 기능의 컴포넌트를 선택하여 사용할 경우 발생하는 오류는 전체 시스템의 생산성과 신뢰성에 큰 악영향을 줄 수 있다. 또한 컴포넌트의 서비스를 이용하려는 개발자는 인터페이스 정보만을 바탕으로 이용에 관한 결정을 해야하며 이러한 결정 시기는 코딩 단계가 아닌 설계, 혹은 분석 단계에서 행해지는 경우가 발생하기도 한다.[7] 따라서, 컴포넌트 명세에 있어서 정형 방법을 사용할 경우 이러한 모호성을 제거함으로써 명세 오류들을 매우 효과적으로 줄여준다. 또한 컴포넌트의 의미적 측면이 수학적 모델링을 통해 정형적으로 명세 되어 컴포넌트가 분석될 수 있다면, 컴포넌트 구현과 사용에 대한 작업을 독립적으로 진행하여 개발 작업의 생산성을 향상시킬 수 있다.

본 논문에서는 컴포넌트 기반 소프트웨어 개발 프로세스의 명세에 있어서 컴포넌트의 품질을 높일 수 있도록 분석력과 논리성이 검증된 정형 명세 언어 Z의 스키마 확장을 이용하여 컴포넌트를 명세하므로써 컴포넌트 구현 및 사용상의 오류를 분석할 수 있는 방법을 제안한다. 2절에서는 관련연구에 대해 기술하고, 3절에서는 정형명세 언어 Z의 스키마 확장을 통한 컴포넌트 명세를 제안하고, 4절에서 결론을 맺는다.

본 연구는 정보통신연구진흥원 2001년도 대학기초연구지원사업 지원에 의하여 연구되었음

2. 관련 연구

2.1 컴포넌트의 정의

부품을 조립해서 제품을 만들어 내는 것처럼 부품화된 소프트웨어들을 조합하여 완성된 소프트웨어를 만들어 낼 수 있는데, 이러한 독립된 단위기능의 소프트웨어 부품을 컴포넌트라고 한다. 컴포넌트는 인터페이스 부분만을 이용하여 개발되는 소프트웨어에 바로 바인딩 시켜 재사용 될 수 있는 독립적인 바이너리 코드이다. 컴포넌트는 범용성, 플랫폼, 추상화, 크기 등에 따라서 여러 형태로 분류될 수 있다.[6] 플랫폼 관점에서 살펴보면 컴포넌트 개발을 지원하는 컴포넌트 모델별로 분류할 수 있다. 현재 소개되어 있는 대표적인 컴포넌트 모델에는 COM, EJB, CORBA 컴포넌트 모델 등이 있다. 따라서 COM모델을 기반으로 만들어진 컴포넌트는 COM 컴포넌트가 되며, EJB 모델을 기반으로 만들어진 컴포넌트는 하나의 EJB Bean이 되며, CORBA 컴포넌트 모델을 기반으로 만들어진 컴포넌트는 CORBA 컴포넌트가 된다.

2.2 컴포넌트 명세 범위 및 방법

컴포넌트 명세는 컴포넌트 기술, 커넥터 기술, 전체 시스템 구성 기술의 세 부분으로 나누어진다. 컴포넌트 기술 부분에서는 컴포넌트에 대한 행위가 기술되고, 커넥터 기술부분에서는 컴포넌트사이의 통신규약의 형태로 기술되며, 전체 시스템 구성 기술 부분에서는 시스템을 이루는 각각의 컴포넌트와 커넥터들을 인스턴스화 시키고 연결하여 실제 시스템을 구성하는 것을 기술한다.[6] 컴포넌트에 대한 기존의 명세 방식은 인터페이스 기술 언어(IDL), 자연어명세, OCL명세가 일반적이다. 그러나 이러한 표기법들은 몇 가지 문제점을 내포하고 있다. 먼저 IDL을 살펴보면, 컴포넌트 사용의 구문 적인 부분만을 기술하며, 오퍼레이션에 대한 의미 또는 제약사항들을 기술하지 않는다.[8] 자연어 명세와 OCL명세는 기본적으로 자연어 문장이나 다이어그램에 기초하기 때문에 부정확함은 물론 해석에 따라 문법적으로 애매함이 존재한다. OCL은 개념언어라기보다는 구현언어에 가깝다. 그래서 표현력은 개념언어에 비해 우수하나, 언어 자체적으로 검증의 능력이 떨어진다. 그래서 컴포넌트 명세를 검증할 수 있는 방법을 제공하지 못한다. 따라서 기존 방식의 컴포넌트 명세 방식이 엄밀성과 정확성이 필수로 요구되는 분야에 적용될 경우 발생될 수 있는 문제들을 완화시키는 해결책으로 정형 방법을 이용한 컴포넌트 명세 기법이 요구된다.

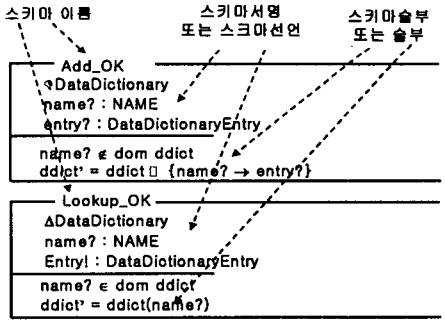
2.3 정형 명세 언어

정형 명세 언어에 관한 연구는 크게 대수적 접근방법과 모델기반 접근방법으로 구분되며, 대수적 접근방법에서는 시스템들을 개별적으로 개발되는 서브시스템으로 구분하여 서브시스템들간의 인터페이스를 정의한다.[6] 모델 기반 접근방법에서는 시스템들을 상태 모델로써 표현한다. 이 상태 모델은 집합이나 함수와 같이 잘 알려진 수학적 개체를 사용하여 구성되고 시스템 모델의 상태에 어떻게 영향을 미치는가를 정의함으로써 명세가 된다.

<표 1> 정형 명세 언어

접근방법	순차적 시스템	병렬 시스템
대 수 적 접근방법	Larch (1985) OBJ (1985)	Lotos (1987)
모델기반 접근방법	Z (1992) VDM (1980)	CSP (1985) Petri Nets (1981)

정형 명세 언어 Z[3]는 집합론 기초한 스키마 구조를 이용하여 시스템을 명세화 한다. Z 명세언어는 일반적으로 상태스키마와 연산 스키마를 포함하고 있으며 이들 스키마는 다른 스키마에 의해 참조될 수 있도록 구조화된다. 또한 연산과 정의를 가능하게 만드는 확장성을 허용하고, 일반적인 정의를 허용한다.

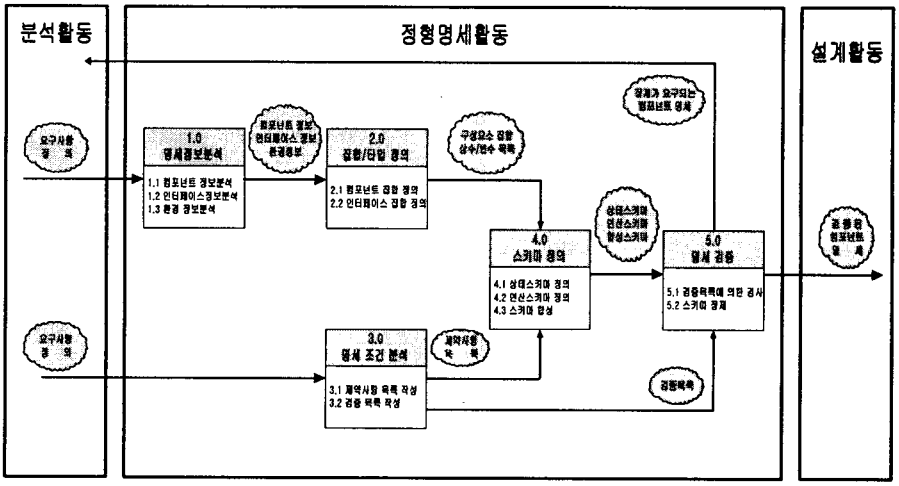


(그림 1) Z 스키마 예

본 논문에서는 모델 기반 명세의 가장 널리 쓰이는 표기법 중 하나인 Z를 이용한 컴포넌트 정형 명세에 대하여 논의한다.

3. 컴포넌트 정형명세를 위한 Z 스키마의 확장

컴포넌트 정형 명세는 수학적 모델을 통해 모호성 및 명세 자체의 불일치를 없애 컴포넌트의 기능을 엄격하게 선언할 수 있다. 본 논문에서는 정형 명세 언어 Z에서 제공하는 스키마를 이용하여 컴포넌트를 요소 정의부, 상수/변수 정의부, 상태스키마 정의부, 연산스키마 정의부로 구성된 스키마로 정의한다. 요소



(그림 2) Z를 이용한 컴포넌트 정형 명세 방법

정의부에서는 컴포넌트의 구성요소들 즉 객체나 다른 컴포넌트들의 집합을 정의하고, 상수/변수 정의부에서는 컴포넌트 내부에서 필요한 상수와 변수를 정의하고, 상태스키마 정의부에서는 컴포넌트의 초기상태를 정의하고, 연산스키마 정의부에서는 컴포넌트의 요소들 사이의 관계 즉 인터페이스를 포함한 컴포넌트의 상호작용과 상태의 변화를 정의한다. [6]에서 제안한 컴포넌트 정형 명세 프로세스(그림2)의 5개의 태스크 중에서 컴포넌트의 상태와 기능을 Z스키마로 정의하는 태스크 4.0 컴포넌트 스키마 정의를 통해 위에서 정의한 스키마 정의를 완성한다. 태스크 4.0 컴포넌트 스키마 정의에서는 컴포넌트와 인터페이스 집합/타입 정의의 산출물인 구성요소 집합과 상수/변수 목록과 명세 조건 분석의 산출물인 제약사항 목록을 입력물로 하여 상태스키마 정의, 연산스키마 정의, 스키마 합성을 통해서 상태스키마, 연산스키마, 합성스키마를 산출한다.

정형 명세 언어 Z는 특정 연산 스키마가 하나의 특별한 상태스키마에 영향을 주는지를 추론하는 것은 모든 연산스키마의 서명(signature)들을 살펴보는 것을 요구하는데 이것은 크고 복잡한 명세에 있어서 실행 불가능하다. 그래서, 하나의 상태스키마를 참조하는 개별적인 연산들을 제거하고 강화된 구조를 통해서 복잡한 명세의 정확성을 향상시키기 위해 다음과 같은 규칙을 통해 정형 명세 언어 Z를 확장하였다.

Speficaition :: = Paragraph
•
•
•
Paragraph

Paragraph :: = BasicTypeDefinition
| AxiomaticDefinition
| GenericDefinition
| AbbreviationDefinition
| FreeTypeDefinition
| Schema
| Component
| Predicate

Component :: = ElementDefinition
| ConstantVariableDefinition
| StateSchemaDefinition
| OperationSchemaDefinition

ElementDefinition :: = ↑(DeclaratinNameList)

ConstantVariableDefinition :: = BasicTypeDefinition
| AxiomaticDefinition
| GenericDefinition
| AbbreviationDefinition
| FreeTypeDefinition

StateSchemaDefinition :: = Schema

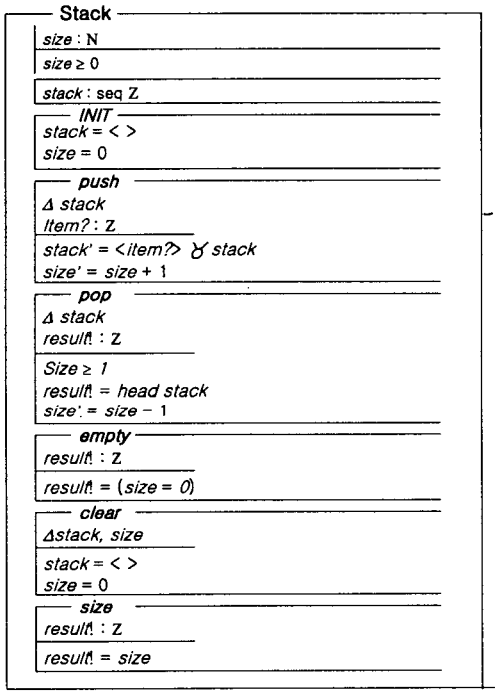
OperationSchemaDefinition :: = Schema

컴포넌트 스키마를 정의하기 위해서 먼저 컴포넌트 각각에 대한 초기 상태스키마를 정의하여야 한다.

```
┌──────── INIT ────────
│ stack = <>
│ size = 0
└──────────────────────
```

(그림 3) Stack의 초기 상태 스키마의 예

다음으로 컴포넌트 연산스키마 정의에서는 내부 연결 규칙과 외부 연결 규칙의 식별에 따라 각각의 컴포넌트 연산스키마를 정의한다. 컴포넌트 내부 연결 규칙이 식별되면 컴포넌트의 상호작용과 관련된 인터페이스를 올바른 연산스키마와 예외처리 연산스키마 각각으로 정의한다. 외부 연결 규칙이 식별되면 컴포넌트들의 집합 사이의 상호작용을 명세 하는 것으로 올바른 연산스키마와 예외처리 연산스키마 각각을 정의해야 한다. (그림 4)와 같이 주어진 집합들과 상태스키마와 연산스키마를 합성하는 것으로 컴포넌트 명세를 완성하는 것을 보인다.



(그림 4) 상태스키마와 연산스키마의 합성의 예

이렇게 정의된 컴포넌트 스키마의 산출물인 상태스키마, 연산 스키마, 합성스키마 명세의 정확성을 검사하기 위해 이전에 작성된 제약사항을 이용하여 검증한다. 이러한 검증을 통해서 정제가 필요한 컴포넌트 명세가 발견되면 이전 단계로의 피드백을 제공하고, 그렇지 않은 경우 검증된 컴포넌트 명세를 설계활동의 기초로 이용한다.

4. 결론

본 논문에서는 소프트웨어 컴포넌트의 품질 향상을 위한 방법으로 Z의 스키마 확장을 통한 컴포넌트 정형명세를 제안하였다. 크고 복잡한 명세에 있어서 정형 명세 언어 Z의 연산스키마가 상태스키마에 어떤 영향을 주는지를 추론 할 수 있도록 하는 것이 불가능하다. 따라서 제안한 스키마의 확장은 강화된 구조를 통하여 복잡한 명세의 정확성을 향상시켰다. 또한 기존의 자연어로써 불명확하게 명세 되던 컴포넌트를 수학적인 모델링을 통해 표현함으로써 애매 모호함이나 불확실성을 최소한으로 줄일 수 있다는 것을 보여주었다. 컴포넌트 개발초기에 명세의 정확성을 검증하여 품질 높은 컴포넌트 개발을 보증함으로써 생산된 컴포넌트의 신뢰성을 크게 증진시킬 수 있다. 그리고, 정형 명세를 이용함으로써 명세 이후의 설계, 구현, 시험 활동 등의 비용을 크게 줄임으로써 전체 개발비용을 절감시킬 수 있다.

향후 연구되어야 할 부분으로는 시스템에 대한 요구사항 명세를 자동화된 방법으로 Z 명세로 변환할 수 있는 시스템 구축에 관한 연구가 필요하다.

[참고문헌]

[1]B.Ratcliff, "Introducting specification using z: A practical Case Study Approach", McGraw-Hill, 1994

[2] S.Stepney, R.Barden, and D. Cooper, "Object Orientation in Z", British Computer Society, 1992

[3] A.Harry, "Formal Methods Fact File: VDM and Z", John Willey & Sons, 1996

[4] M.Vaziri, D.Jackson, "Some Shortcomings of OCL, the Object Constraint Language of UML", Response to OMG's Request for Information on UML 2.0, 1999

[5]A.Parrish, B.Dixon and D.Hale, "Component Based Software Engineering: A Broad Based Model is Needed", ICSE, 1999

[6] 장종표 "컴포넌트 품질향상을 위한 정형명세 프로세스", 2002. 2 전남대학교 박사학위 논문

[7] 서동수, "정형기법에 의한 재사용 컴포넌트 및 인터페이스 명세 기술 연구", 정보처리학회논문지, 제8-D권 제1호, 2001

[8] 박찬진, 우치수, "컴포넌트 정형명세", 정보처리학회지, 제7권 제4호, 2000

[9] Roger S. Pressman, "소프트웨어 공학 실무적 접근", McGraw-Hill Korea, 2001