

UML 다이어그램의 검증을 위한 검증 시스템의 설계

김진수^U

김재웅^{**}

김치수^{***}

*건양대학교 정보전자통신공학부, **공주대학교 정보과학과, ***공주대학교 전자계산학과
jinskim@kvtis.konyang.ac.kr

A Design of the Verifying System for verifying UML Diagram

Jin-Soo Kim^U Jae-Wung Kim^{**} Chi-Su Kim^{***}

* Dept. of Information, Electronics and Communication Eng., Konyang University

** Dept. of Information & Science, Kongju University

*** Dept. of Computer Science, Kongju University

요 약

최근 객체지향 설계에 UML을 많이 사용하게 되면서 개발자들은 작성된 UML 다이어그램들이 일관성이 있고 완전한가를 검증할 필요성이 대두되기 시작하였다. 본 논문에서는 [1]에서 제공된 다이어그램의 일관성과 완전성 검증 규칙을 사용하여 작성된 UML 다이어그램의 일관성과 완전성을 검증하기 위한 검증 시스템을 설계하였다. 검증 시스템의 설계에는 UML 다이어그램을 활용하였으며 프레임을 이용하여 검증 시스템의 다이어그램에 대한 내부표현을 제시하였다.

1. 서 론

최근에 객체지향 방법이 소프트웨어를 개발하는 탁월한 패러다임이 되고 있고 객체지향 방법이 기존의 개발 방법들에 비해 개발되는 소프트웨어의 품질을 향상시키고 생산성을 증가시킨다는 것은 이미 많이 알려져 있다. 일반적으로 크고 복잡한 소프트웨어 시스템은 커다란 다이어그램의 집합으로 구성되지만 이들 각각의 다이어그램들이 일관성이 있고 완전한가를 알기는 매우 어렵다. 일반적으로 다이어그램들의 일관성과 완전성을 보장하게 만드는 일은 다음과 같은 이유들로 인해 상당히 어려운 일이다. 첫째, 객체지향 설계는 개발자의 논리적 사고의 결과이기 때문에, 크고 복잡한 소프트웨어 시스템을 다른 관점과 다른 추상화 수준에서 개발자의 생각을 일관성이 있고 완전하게 유지한다는 것은 어려운 일이다. 둘째, 크고 복잡한 소프트웨어 시스템의 객체지향 설계는 보통 설계팀의 생산품이다. 셋째, 객체지향 설계는 반복적인 작업이다. 설계의 변경이 발생하는 동안 설계자들이 이 모든 변화를 서로간에 일관성이 있고 많은 다이어그램들간에 완전하게 유지하기는 힘들다[1].

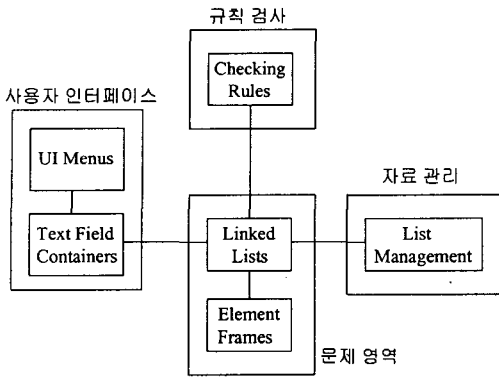
이러한 문제를 해결하기 위하여 본 논문에서는 최근 객체지향 개발에서 많이 사용되고 있는 UML 다이어그램에 대한 일관성과 완전성을 검사하기 위하여 [1]에서 제안된 검증 규칙들을 이용하여 다이어그램의 일관성과 완전성을 검증할 수 있는 검증 시스템을 설계하였다.

2장에서는 UML 다이어그램을 사용하여 검증 시스템을 설계하고 3장에서는 프레임을 이용하여 검증 시스템의 다이어그램에 대한 내부표현을 보이고 4장에서 결론을 맺는다.

2. 검증 시스템의 설계

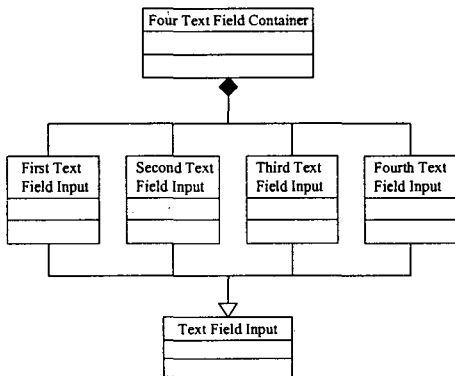
2.1 검증 시스템의 구성

[1]에서의 규칙을 기반으로 다이어그램들간의 일관성과 완전성을 검사하기 위하여 본 논문에서 제안한 검증 시스템은 다음 (그림 1)과 같이 크게 4개의 서브 시스템들로 구성된다. 첫 번째는 사용자 인터페이스 서브시스템으로 Text_Field_Containers와 UI_Menus로 구성되는데 Text_Field_Containers는 각 항목이 텍스트 필드를 갖는 텍스트 필드의 항목들의 집합이라고 할 수 있고 UI_Menus는 사용자의 입력을 도와주는 기능들을 제공한다. 두 번째는 문맥영역 서브시스템으로 Linked_Lists와 Element_Frames로 구성되는데 Linked_Lists는 single-linked-list를 생성하기 위해서 Element_Frames의 서비스를 이용하고 Text_Field_Containers로부터 받은 정보를 single-linked-list의 요소에 저장한다. 세 번째는 자료관리 서브시스템으로 그 요소인 List_Management는 single-linked-list를 외부 파일로 저장하기도 하고 저장된 파일로부터 다시 생성하기도 한다.



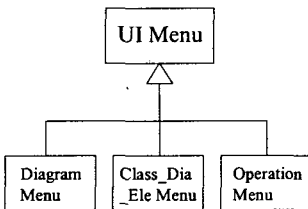
(그림 1) 검증 시스템의 구성

마지막 서브시스템은 규칙 검사 서브시스템으로 그 요소인 Rule-Based-Checking은 single-linked-list로부터 자료를 얻기 위해서 Linked_Lists의 서비스를 이용한다.



(그림 2) Text_Field_Containers에 대한 클래스 다이어그램

Text_Field_Containers는 (그림 2)와 같이 네 종류의 텍스트 필드 항목들을 가지고 있으며 프레임의 슬롯에 값을 넣는 것에 따라 종류가 결정된다.



(그림 3) UI_Menu에 대한 클래스 다이어그램

사용자 인터페이스 서브시스템의 UI_Menu는 (그림 3)에서와 같이 다이어그램 메뉴, 다이어그램 요소 메뉴, 오퍼레이션 메뉴와 같은 클래스들로 구성

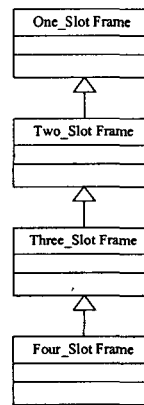
되어 있다.

문제 영역 서브시스템의 Linked_Lists는 (그림 4)와 같이 SL_List 클래스와 SL_List_Element 클래스로 구성되어 있다.



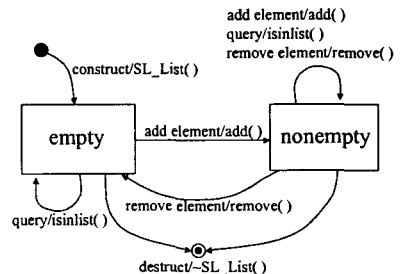
(그림 4) Linked_Lists에 대한 클래스 다이어그램

문제 영역 서브시스템의 Element_Frame은 (그림 5)와 같이 네 클래스로 구성되어 있다.



(그림 5) Element_Frame에 대한 클래스 다이어그램

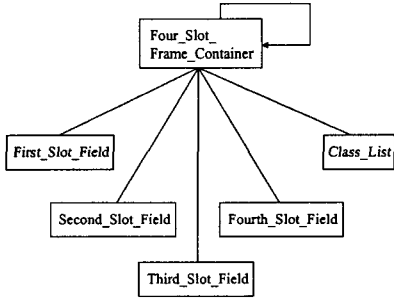
SL_List 클래스는 (그림 6)과 같이 empty와 non-empty의 두 상태를 가지는데 클래스 생성자 함수인 SL_List()가 수행되면 비어있는 single-linked-list가 생성되고 그 상태는 empty 상태가 된다.



(그림 6) SL_List 클래스에 대한 상태 다이어그램

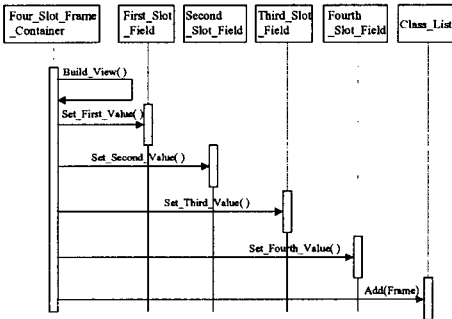
먼저 다이어그램 메뉴에서 클래스 다이어그램이 선택되고, 다이어그램 요소 메뉴에서 클래스 다이어그램의 요소인 relationship이 선택되고, 오퍼레이션 메뉴에서 오퍼레이션 add()가 선택되면, 네 개의 슬롯을 갖는 하나의 프레임이 생성된다. 먼저 사용자로부터 관련성의 이름을 입력받아 프레임의 처음 슬롯에 넣고 다음으로 관련성의 subset 이름을 받아 프레임의 두 번째 슬롯에 넣는다. 관련성이 시작되

는 클래스의 이름을 입력받아 관련성 프레임의 세 번째 슬롯에 넣고 마지막으로 관련성이 종료되는 클래스의 이름을 입력받아 관련성 프레임의 네 번째 슬롯에 넣으면 이 프레임은 single-linked-list인 ClassList에 하나의 요소로 추가된다. 다음 (그림 7)은 이러한 시나리오를 보여주는 객체 다이어그램이다.



(그림 7) 관련성을 추가하는 객체 다이어그램

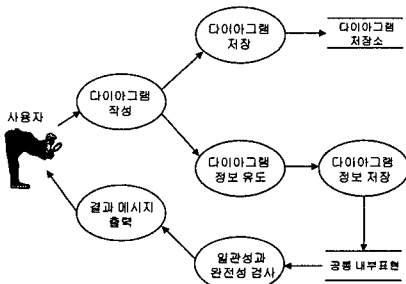
ClassList 리스트에 하나의 관련성을 추가하는 시나리오는 다음 (그림 8)과 같이 순차 다이어그램으로도 나타낼 수 있다.



(그림 8) 클래스를 추가하는 순차 다이어그램

2.2 검증 시스템의 검증 과정

다음 (그림 9)는 검증 시스템의 자료흐름도를 보여주고 있다.



(그림 9) 검증 시스템의 자료흐름도

본 논문에서 설계한 검증 시스템은 먼저 사용자가 다이어그램을 선택할 수 있는 다이어그램 메뉴, 다이어그램 요소를 선택할 수 있는 다이어그램 요소 메뉴, 이러한 다이어그램 요소의 오퍼레이션을 선택할 수 있는 메뉴를 제공한다. 사용자가 다이어그램, 다이어그램 요소, 오퍼레이션을 선택하면 검증 시스템은 이러한 다이어그램의 요소에 대한 필요한 정보를 가져와서 공통 내부표현으로 저장할 수 있도록 한다. 이렇게 저장된 공통 내부표현을 이용하여 일관성과 완전성에 대한 검사를 수행하며 수행된 결과는 사용자에게 메시지로 제공될 수 있다.

3. 프레임을 이용한 검증 시스템의 내부표현

일반적으로 프레임은 값을 넣을 수 있는 여러 개의 슬롯으로 구성되는데 타입의 값들을 나타내기 위하여 사용되기도 한다. 예를들어 [1]에서 제시된 Relationship 타입의 각 값은 관련성들간에 구별될 수 있는 유일한 이름을 갖고 있는 하나의 관련성이라고 할 수 있다. 따라서 Relationship 타입에 대한 프레임은 Relationship-Name이라는 슬롯을 가지는데 이 슬롯의 값은 Relationship 타입에서 유일하다. 관련성이 어떠한 부분 집합을 가지는가를 나타내기 위하여 Relationship 타입에 대한 프레임은 SubsetName이라는 슬롯을 가지게 되고 이 슬롯에 부분 집합 이름이 들어가면 이 구체적인 관련성은 해당 부분 집합의 원소가 된다는 것을 나타낸다.

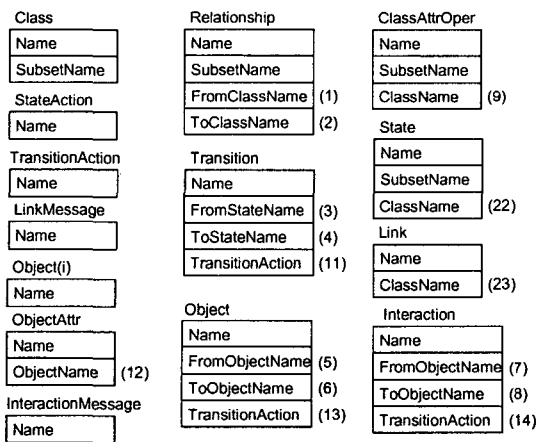
타입은 함수의 정의역이나 치역의 타입을 나타낼 수 있다. 예를들어, [1]에서 Relationship 타입이 RfC^1 함수와 RtC^2 함수의 정의역 타입으로 사용되었기 때문에 Relationship 타입에 대한 프레임은 추가적으로 FromClassName과 ToClassName의 두 슬롯을 가지게 된다. FromClassName 슬롯은 Class 타입이 RfC^1 함수의 치역이므로 관련성이 시작되는 클래스의 이름이 들어가고 ToClassName 슬롯에는 RtC^2 함수의 치역이므로 관련성이 종료되는 클래스의 이름이 들어가게 된다.

이상과 같이 Relationship 타입에 대한 프레임은 RelationshipName, SubsetName, FromClassName, ToClassName의 네 개의 슬롯으로 구성되어 있다. 다음 (그림 10)은 Relationship 타입에 대한 슬롯을 보여주고 있다.

RelationshipName
SubsetName
FromClassName
ToClassName

(그림 10) Relationship 타입에 대한 슬롯

이와 유사하게 각 타입들에 대한 프레임을 얻을 수 있다. 다음 (그림 11)은 각 타입들에 대한 프레임을 보여주고 있다.



(그림 11) 각 타입에 대한 프레임들

(그림 11)에서 각 슬롯의 우측에 기재된 번호는 [1]에 제시된 R의 원소인 함수에 붙여진 번호로써 해당 함수의 치역 타입의 값을 가진다는 의미이다. 결국 각 타입들은 single-linked-list 형태로 표현될 수 있으며 다음 (그림 12)는 (그림 2), (그림 3), (그림 4), (그림 5)의 클래스 다이어그램에서 유도된 정보를 가지고 있는 검증 시스템의 공통내부 표현이라고 할 수 있고 이러한 내부표현들을 입력으로 일관성과 완전성을 검증할 수 있다. 객체 다이어그램, 상태 다이어그램, 순차 다이어그램에 대한 공통내부 표현도 (그림 11)의 프레임을 사용하여 (그림 12)와 같이 표현될 수 있다.

4. 결론

본 논문에서는 [1]에서 제공된 다이어그램의 일관성과 완전성 검증 규칙을 사용하여 작성된 UML 다이어그램의 일관성과 완전성을 검증하기 위한 검증 시스템을 설계하였다. 검증 시스템의 설계에는 UML 다이어

그램을 활용하였으며 프레임을 이용하여 검증 시스템의 다이어그램에 대한 내부표현을 제시하였다.

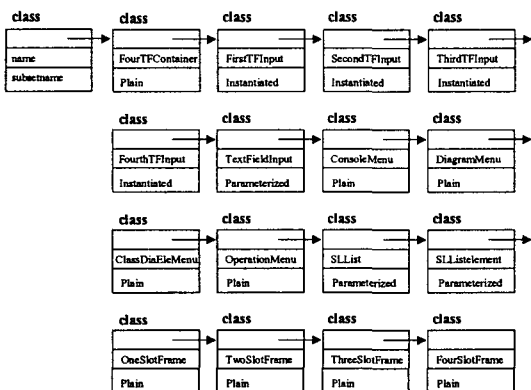
본 논문에서 설계된 검증 시스템은 다이어그램에서 유도된 정보를 기반으로 일관성과 완전성을 검증하는데 필요한 다이어그램의 정보를 공통적인 내부표현으로 저장할 수 있도록 하기 위하여 프레임을 이용하였고 이러한 검증과정을 보이기 위하여 다이어그램의 일관성과 완전성을 검증할 수 있는 검증 시스템에 대한 각 다이어그램을 대상으로 정보를 유도하였고 유도된 정보를 기반으로 검증에 기초가 되는 공통 내부 표현을 프레임을 이용하여 설계하는 과정을 제시하였다.

설계된 검증 시스템은 다양한 객체지향 다이어그램에 대하여 적용이 가능하며 기존의 다이어그램 도구와도 결합이 가능하도록 구성되어 있다.

향후연구과제로는 객체지향 소프트웨어 개발을 좀 더 효율적으로 할 수 있도록 UML의 모든 다이어그램을 처리할 수 있는 그래픽 컴파일러와 같은 도구의 개발이 요구된다.

참고 문헌

- [1] 김재용 외, "UML 다이어그램간의 일관성과 완전성을 위한 검증 규칙 생성에 관한 연구", 한국멀티미디어학회 논문지, 제3권, 제3호, 290-298, 2000
- [2] Paul Hramon and Mark Watson, Understanding UML : The Developer's Guide, Morgan Kaufmann Publishers, Inc., 1997
- [3] Edwards, J. M. and Henderson-Sellers, B., "A graphical notation for object-oriented analysis and design", Journal of Object-Oriented Programming 4(9) , 53-74, 1994
- [4] Robert H. Bourdeau and Betty H. Cheng, "A Formal Semantic for Object Model Diagrams", IEEE Trans. on Software Engineering, Vol. 21, No. 10, pp. 799-821, 1995



(그림 12) 검증 시스템에 대한 공통 내부 표현