UML 다이아그램의 일관성 및 완전성 검증 규칙

김재웅*○ 김진수** 황선명*
*대전대학교 컴퓨터공학과, **건양대학교 정보전자통신공학부
{jykim, sunhwang}@zeus.taejon.ac.kr, jinskim@kytis.konyang.ac.kr

The Rules for Verifying Completeness and Consistency of the Unified Modeling Language Diagrams

Jae-Wung Kim*○ Jin-Soo Kim** Sun-Myung Hwang*
* Dept. of Computer Engineering, Taejon University
** Dept. of Information, Electronics and Communication Eng., Konyang University

요 약

본 논문에서는 최근 객체지향 설계에 많이 사용되고 있는 UML 다이아그램들의 일관성과 완전성을 해결하기 위하여 UML의 각 다이아그램들을 ER 모델로 표현하고, 각 다이아그램에 대한 공통된 표현으로 통합된 ER 다이아그램을 제공하며 일련의 집합과 함수들을 사용하여 정형적으로 명세한 다음 이러한 정형 명세를 기반으로하여 일관성과 완전성 검사를 수행하기 위한 규칙을 제공한다. 이 규칙은 다이아그램의 구문(syntax)과 의미(semantic)에 대하여 모두 검사할 수 있게 하기 위하여 구문 완전성, 구문 일관성, 의미 완전성, 의미 일관성으로 나누어 생성한다. 이렇게 생성된 규칙들은 추후 CASE 도구에 포함되어 다이아그래밍의 일관성 및 완전성을 검사할 수 있는 CASE 도구로 발전할 수 있게 된다.

1. 서론

최근에 객체지향 방법이 소프트웨어를 개발하는 탁월한 파라다임이 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 다이아그램의 집합으로 구성되지만 이들 각각의 다이아그램들이 일관성이 있고 완전한가를 알기는 매우 어렵다. 이러한 문제를 해결하기 위하여 본 논문에서는 최근 객체지향 개발에서 많이 사용되고 있는 UML 다이아그램들에 대한 일관성과 완전성을 검사하기 위하여 각 다이아그램들을 ER 모델로 표현하고, 각 다이아그램에 대한 공통된 표현으로 통합된 ER 다이아그램을 제공한다. 이렇게 통합된 다이아그램은 일련의 집합과 함수들을 사용하여 정형적으로 명세되며, 이러한 정형 명세를 기반으로하여 일관성과 완전성 검사를 수행하기 위한 규칙을 제공한다. 이 규칙은 추후 CASE 도구에 포함되어 다이아그래밍의 일관성과 완전성을 검사할 수 있는 CASE 도구로 발전할 수 있게 된다.

2장에서는 본 논문과 관련된 연구들을 소개하였고, 3장에서는 UML 다이아그램에 대한 ER 다이아그램들을 제공하고 4장에서는 UML 다이아그램의 일관성과 완전성 규칙을 제시하고 5장에서 결론을 맺는다.

2. 관련 연구

2.1 다이아그램의 일관성과 완전성 문제

일반적으로 다이아그램들의 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. 세째, 객체지향 설계는 반복적인 작업이다. 설계의 변경이 발생되는 동안 설계자들이 이 모든 변화를 서로간에 일관성이 있고 많은 다이아그램들간에 완전하게 유지하기는 힘들다.

2.2 UML 다이아그램

본 논문에서는 최근 객체지향 개발 방법에서 표준화로 자리잡아 가고 있는 UML(Unified Modeling Language) 다이아그램중에서 클래스 다이아그램, 객체 다이아그램, 순차 다이아그램, 상태 다이아그램을 대상으로 다이아그램들간의 일관성과 완전성을 검사하도록 한다.

3. UML 다이아그램의 ER 모델

본 논문에서는 2.2에서 제시된 각 다이아그램들을 ER 다이아그램으로 표현한 다음 일관성과 완전성을 검사하기 위하여 하나의 ER 모델로 통합한다.

들간의 관련성이다.

(1) 클래스 다이아그램에 대한 ER 다이아그램

클래스 다이아그램의 필수적인 요소는 클래스와 클래스

들간의 관련성이다.

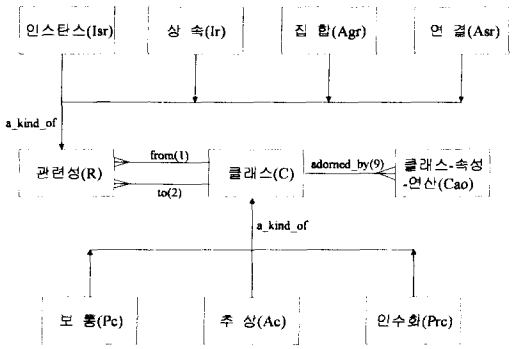

그림 1. 클래스 다이아그램에 대한 ER 다이아그램

(2) 객체 다이아그램에 대한 ER 다이아그램
객체 다이아그램의 필수적인 요소는 객체와 그들의 관련성이다.

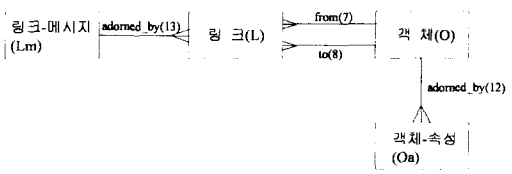

그림 2. 객체 다이아그램에 대한 ER 다이아그램

(3) 순차 다이아그램에 대한 ER 다이아그램
순차 다이아그램의 필수적인 요소는 객체와 그들간의 상호작용이다.

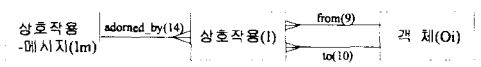

그림 3. 순차 다이아그램에 대한 ER 다이아그램

(4) 상태 다이아그램에 대한 ER 다이아그램
상태 다이아그램의 필수요소는 상태와 상태전이이다. 다음 그림들은 각 다이아그램에 대한 ER 다이아그램이다.

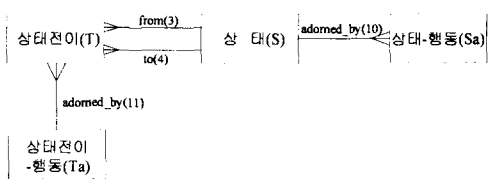

그림 4. 상태 다이아그램에 대한 ER 다이아그램

(5) 통합된 ER 다이아그램
다이아그램들의 일관성과 완전성 검사를 위하여 각 ER 다이아그램들을 하나의 ER 다이아그램으로 통합하면 그림 5.와 같게 된다.

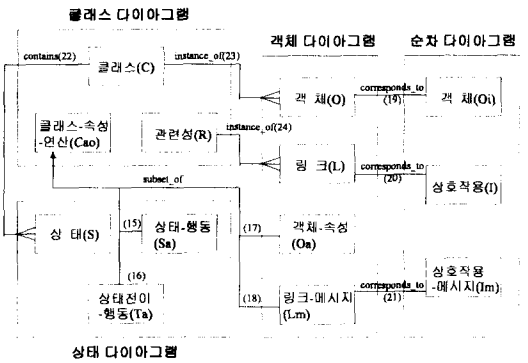

그림 5. 통합된 ER 다이아그램

4. UML 다이아그램의 일관성 및 완전성 검사 규칙

3장에서 구축된 다이아그램에 대하여 집합과 함수를 이용하여 정형적으로 명세한 다음 함수들의 의미에 따라 다이아그램의 일관성과 완전성을 보장하기 위한 일련의 규칙들을 유도하고 이들 규칙에 정형 명세를 주고자 한다. 이들 다이아그램과 규칙들의 정형 명세는 수학적 기호로 표현된다. 그림 5.의 통합된 ER 다이아그램은 일련의 엔티티 타입과 이들 엔티티 타입들간의 관련성으로 구성된다. 따라서 이 다이아그램을 E와 R의 두 튜플로 정의할 수 있다.

UMLOOD = < E, R >
E : 일반화된 엔티티 타입들의 집합
R : 일반화된 엔티티 타입들간의 관련성의 집합

통합된 ER 다이아그램에서 각 일반화된 엔티티 타입은 집합 E의 한 요소이다. 따라서 첫 번째 투플 E는 다음과 같은 일반화된 엔티티 타입의 집합을 이용하여 정형적으로 정의된다.

$$E = \{C,R,Cao,S,T,Sa,Ta,O,L,Oa,Lm,I,Im\} \quad <1>$$

여기에서 제시된 집합 E의 요소인 엔티티 타입의 집합들은 추상자료형 집합을 사용하여 정형적으로 명세될 수 있다.
UMLOOD의 정형화된 정의의 두번째 투플은 R이다. 투플 R은 다음과 같은 함수들로 정형적으로 정의된다.

$$R = \{RfC^{1}, RtC^{2}, TfS^{3}, TtS^{4}, LfO^{5}, LtO^{6}, IfOi^{7}, ItOi^{8}, CaoC^{9}, SaS^{10}, TTa^{11}, OaO^{12}, LLm^{13}, IIm^{14}, SaCao^{15}, TaCao^{16}, OaCao^{17}, LmCao^{18}, OiO^{19}, IL^{20}, ImLm^{21}, ScC^{22}, OcC^{23}, LcR^{24}\} \quad <2>$$

집합 R의 각 요소는 ER 다이아그램에서 엔티티 타입들간의 관련성을 설명하는 함수들로 정의된다.
본 논문에서의 일관성과 완전성은 UML 다이아그램의 모든 필수적인 요소들이 UML에서 사용되는 구문과 의

미를 유지하고 있는지를 확인하는데 사용된다. UML의 구문을 유지하는 일관성을 구문 일관성, 의미를 유지하는 일관성을 의미 일관성이라고 한다. 또한 UML의 구문적 요구사항을 만족하는 완전성을 구문 완전성, UML의 의미적 요구사항을 만족하는 완전성을 의미 완전성이라고 한다.

<정의> UML의 다이아그램이 구문적으로 의미적으로 일관성이 있고 완전하려면 <1>에 정의된 모든 집합의 모든 요소들이 <2>에 정의된 모든 함수들을 만족할 수 있어야 한다

(1) 구문 완전성(syntactic completeness)을 위한 규칙

규칙(11) 어떠한 클래스도 클래스 다이아그램에서 단독으로 존재할 수 없다

$\forall c(c\in C\rightarrow \exists r(r\in R\wedge(RfC^{-1}(c)=r\vee RtC^{-1}(c)=r)))$

규칙(12) 어떠한 객체도 객체 다이아그램에서 단독으로 존재할 수 없다

$\forall o(o\in O\rightarrow \exists l(l\in L\wedge(LfO^{-1}(o)=l\vee LtO^{-1}(o)=l)))$

규칙(13) 어떠한 객체도 순차 다이아그램에서 단독으로 존재할 수 없다

$\forall oi(oi\in Oi\rightarrow \exists i(i\in I\wedge(IfOi^{-1}(oi)=i\vee ItOi^{-1}(oi)=i)))$

규칙(14) 어떠한 상태도 상태 다이아그램에서 단독으로 존재할 수 없다

$\forall s(s\in S\rightarrow \exists t(t\in T\wedge(TfS^{-1}(s)=t\vee TtS^{-1}(s)=t)))$

규칙(111) 클래스 관련성의 양쪽 끝은 반드시 클래스들과 연결되어야 한다

$\forall r(r\in R\rightarrow \exists !c_i,c_j((c_i\in C\wedge c_j\in C)\wedge (RfC(r)=c_i\wedge RtC(r)=c_j)))$

규칙(112) 객체 링크 관련성의 양쪽 끝은 반드시 객체들과 연결되어야 한다

$\forall l(l\in L\rightarrow \exists !o_i,o_j((o_i\in O\wedge o_j\in O)\wedge (LfO(l)=o_i\wedge LtO(l)=o_j)))$

규칙(113) 객체 상호작용 관련성의 양쪽 끝은 반드시 객체들과 연결되어야 한다

$\forall i(i\in I\rightarrow \exists !oi_i,oi_j((oi_i\in Oi\wedge oi_j\in Oi)\wedge (IfOi(i)=oi_i\wedge ItOi(i)=oi_j)))$

규칙(114) 상태전이 관련성의 양쪽 끝은 반드시 상태들과 연결되어야 한다

$\forall t(t\in T\rightarrow \exists !s_i,s_j((s_i\in S\wedge s_j\in S)\wedge (TfS(t)=s_i\wedge TtS(t)=s_j)))$

(2) 구문 일관성(syntactic consistency)을 위한 규칙

규칙(21) 인스탄스 관련성의 꼬리는 파라메타화된 클래스에만 연결되어 있어야 한다

$\forall isr(isr\in Isr\rightarrow \exists !prc(prc\in Prc\wedge RfC(isr)=prc))$

(3) 의미 완전성(semantic completeness)을 위한 규칙

규칙(31) 하나의 클래스는 객체 다이아그램에서 적어도 하나의 객체를 포함한다

$\forall c(c\in C\rightarrow \exists o(o\in O\wedge OcC^{-1}(c)=o))$

(4) 의미 일관성(semantic consistency)을 위한 규칙

규칙(41) 하나의 객체는 클래스의 한 인스탄스이다.

$\forall o(o\in O\rightarrow \exists !c(c\in C\wedge OcC(o)=c))$

규칙(42) 하나의 상태는 클래스의 한 상태이다

$\forall s(s\in S\rightarrow \exists !c(c\in C\wedge ScC(s)=c))$

규칙(43) 하나의 링크는 클래스 관련성의 한 인스탄스이다

$\forall l(l\in L\rightarrow \exists !asr(asr\in Asr\wedge LcR(l)=asr))$

규칙(411) 하나의 객체 속성은 클래스의 한 속성이다

$\forall oa(oa\in Oa\rightarrow \exists !cao(cao\in Cao\wedge OaCao(oa)=cao))$

규칙(412) 하나의 상태 행동은 클래스의 한 연산이다

$\forall sa(sa\in Sa\rightarrow \exists !cao(cao\in Cao\wedge SaCao(sa)=cao))$

규칙(413) 하나의 상태전이행동은 클래스의 한 연산이다

$\forall ta(ta\in Ta\rightarrow \exists !cao(cao\in Cao\wedge TaCao(ta)=cao))$

규칙(414) 하나의 링크 메시지는 클래스의 한 연산이다

$\forall lm(lm\in Lm\rightarrow \exists !cao(cao\in Cao\wedge LmCao(lm)=cao))$

규칙(415) 순차 다이아그램에서 한 객체는 객체 다이아그램에서의 한 객체이다

$\forall oi(oi\in Oi\rightarrow \exists !o(o\in O\wedge OiO(oi)=o))$

규칙(416) 순차 다이아그램에서 하나의 상호작용은 객체 다이아그램에서 하나의 링크이다

$\forall i(i\in I\rightarrow \exists !l(l\in L\wedge IL(i)=l))$

규칙(417) 순차 다이아그램에서 하나의 상호작용 메시지는 객체 다이아그램에서 하나의 링크 메시지이다

$\forall im(im\in Im\rightarrow \exists !lm(lm\in Lm\wedge ImLm(im)=lm))$

5. 결론

본 논문에서는 UML 다이아그램들간의 일관성 및 완전성 검사를 위한 규칙들을 유도하였다. 이러한 규칙들을 유도하기 위하여 UML의 각 다이아그램들을 일련의 집합과 함수들로 정형적으로 정의하였으며 이 함수들의 의미에 기반하여 다이아그램들의 구문과 의미에 따라 일관성과 완전성을 보장하기 위한 일련의 규칙들을 유도하였다. 이들 집합, 함수, 규칙들은 모두 수학적인 기호로 정형적으로 명세되므로 추후 CASE 도구에 포함되어 다이아그래밍의 일관성과 완전성을 검사할 수 있는 CASE 도구로 발전할 수 있다.

6. 참고문헌

[1] Paul Hramon and Mark Watson, Understanding UML : The Developer's Guide, Morgan Kaufmann Publishers, Inc., 1997

[2] Martin Fowler, UML Distilled, Addison Wesley, 1997

[3] Edwards, J. M. and Henderson-Sellors, B., "A graphical notation for object-oriented analysis and design", Journal of Object-Oriented Programming 4(9) , 53-74, 1994

[4] 박진호, 김수동, 류성열, "UML 다이아그램들 간의 일관성 검증 방법", 한국정보과학회 '98 봄학술발표논문집, 제25권, 제1호, 524-526, 1998

[5] 정기원, 조용선, 권성구, "객체지향 설계방법에서 오류 검출과 일관성 점검기법", 한국정보처리학회 논문지, 제6권, 제8호, 2072-2087, 1999

[6] 김도형, 정기원, "객체지향 분석과정에서 오류와 일관성 점검방법", 정보과학회논문지(B), 제26권, 제3호, 380-391, 1999