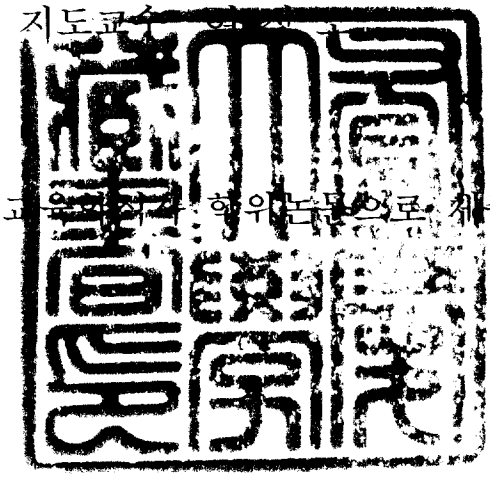
교육학석사 학위논문

UML 활동 다이어그램의 EMFG 표현에 관한 연구

지도교수 [illegible]



이 논문을 교육학석사 학위논문으로 제출함

2005년 8월

부경대학교 교육대학원

전 산 교 육 전 공

이 미 순

이미순의 교육학석사 학위논문을 인준함

2005년 6월 17일

주 심 이학박사 윤 성 대 

위 원 이학박사 박 홍 복 

위 원 공학박사 여 정 모 

[차 례]

[표 차례]

[그림 차례]

The Study on the UML Activity Diagram Representation with the EMFG

Mi Soon Lee

Graduate School of Education

Pukyong National University

Abstract

Lately, it is a trend that all the corporations are broadly using, object-oriented UML when they develop the information system in accordance with the development of object-oriented technology.

They actively use UML activity diagram because of the user-friendliness, object-oriented standard observance, event and the dynamic flow of corporation's activity among the elements of UML. Although UML has a good quality in graphing and user interface, UML is lack of the analytic ability. This paper suggests transforming UML activity diagram into EMFG(Extended Mark Flow Graph) to make workflow modeling and quality analysis easy. It also suggests the method of verifying logical error and the possibility of performance. This discovers the possibility of deadlock and of pre-error of workflow. We can save time and cost by analyzing dynamically through the performance of EMFG Simulation. We can increase reusability by adding or modifying partly in the use of EMFG, too.

Ⅰ. 서 론

최근의 급속한 경영환경 변화 아래서 생존하기 위해 각 기업들은 프로세스 혁신(Process Innovation)이나 BPR(Business Process Reengineering)등을 통해 경쟁력을 확보하려 총력을 기울이고 있다[13]. 기업의 경쟁력을 확보하기 위한 프로세스 혁신이나 BPR의 구현을 성공적으로 수행하기 위해 많은 기업에서 워크플로우 관리 시스템의 도입이 확산되고 있는 추세이다[9]. 용어 자체에 이미 프로세스 자동화의 개념이 내포되어 있어 워크플로우관리시스템을 BPR성공을 위한 핵심정보 기술이라고 평가하고 있다[3].

워크플로우를 분석하기 위해서는 우선 업무의 동적인 구조를 정확히 표현하는 워크플로우 모델이 작성되어야 하는데, 지금까지 워크플로우를 모델링하는 데 주로 사용된 도구로는 IDEF3(ICAM Definition3), ARIS EPC(Event driven Process), 페트리넷, EMFG 등을 들 수 있다[6-8].

최근에는 객체지향기술의 발달에 따라 정보 시스템의 개발에 있어 객체지향 모델링 표준인 UML이 광범위하게 사용되는 추세이다. UML 요소 중의 하나인 활동 다이어그램(AD; Activity Diagram)은 기업 활동의 동적 흐름표현, 이벤트 표현, 객체지향 표준준수 및 사용자 편이성이라는 측면에서 기업 프로세스를 모델링 하는데 적합한 방법으로 인식되어 활발히 사용되는 추세이나 분석능력 부족이라는 단점이 있다. 이를 해결하기 위한 방법으로 수학적 형식론에 기반하여 시스템의 행위를 수학적으로 기술할 수 있는 페트리 넷을 이용한 연구가 진행되고 있다[9]. UML 활동다이어그램의 페트리 넷의 변환에서는 UML AD와 1대 1 매핑이 불가능하여 페트리 넷에 복수개의 블록을 도입하여 변환한다. 블록의 도입은 페트리 넷으로 변환된 그래프에 복잡성을

가중하게 되며, 또한 페트리 넷으로 변환하여 도달성 트리를 이용한 분석은 직관적인 검증을 어렵게 한다.

본 논문에서는 워크플로우의 모델링 및 정성적 분석을 쉽게 하기 위해, 개념적 설계와 시각적 표현이 가능하고 정형적이며 동적 특성을 분석할 수 있는 EMFG(Extended Mark Flow Graph)를 이용하여 UML AD를 표현하는 방법을 제안한다. UML AD를 EMFG로 표현하기 위하여 먼저 UML AD의 구성요소를 각 활동이나 분기 흐름을 나타내는 요소들을 구분하고, 상태변화에 따라 UML AD의 구조를 구분하여 정의하고 각 구조에 따른 EMFG 표현 방법을 제안하고자 한다. 이에 따라 UML AD를 표현한 EMFG에 대해 수학적 알고리즘을 적용한 EMFG 시뮬레이션을 통해 직관적인 마크의 흐름으로 워크플로우의 흐름을 파악하고, EMFG의 점화벡터순서를 나타내는 마크벡터를 이용하여 또한 마크벡터를 분석하여 UML AD로 모델링한 워크플로우가 수행완료 가능한지를 분석할 수 있다.

본 논문은 먼저, 워크플로우, UML, EMFG에 대한 관련연구를 알아보고 UML AD를 EMFG로 표현하기 위한 용어들을 정의한다. 그리고 UML AD의 각 요소의 특징을 알아보고 상태변화에 따른 구조를 정의한 후 UML AD의 각 요소들과 각 구조에 따라 EMFG로 표현하는 방법을 제시한다. UML AD의 수행완료 가능성을 분석하기 위해 먼저 수행완료가능성을 정의하고, UML AD를 표현한 EMFG를 동작시키기 위한 초기마크를 구한다. 수행완료가능성 알고리듬에 따라 UML AD를 EMFG로 표현한 후 이를 시뮬레이션을 수행하여 수행동작을 알아보고 끝으로 수행된 시뮬레이션 결과를 통해 UML AD의 수행완료가능성을 분석한다.

Ⅱ. 관련연구

본 장에서는 워크플로우의 개념과 UML 및 UML AD에 대하여 알아본 후 EMFG의 개념 및 동작과 특징에 대해 알아본다.

2.1 워크플로우

워크플로우는 사전에 정의된 절차와 규칙에 따라 기업 또는 조직체내 또는 조직체 간에 발생하는 일련의 업무 흐름을 말한다. 즉 프로세스를 정의하고, 업무가 관련 정보와 함께 정해진 시간 안에 자동적으로 수행하는 것을 말한다[15].

초기의 워크플로우는 종이 없는 사무실을 구현하는 것이 목적이었으나 최근의 워크플로우는 기업, 공장, 정부 단체 등의 거의 모든 업무를 지원하며 조직 내 또는 조직 간의 핵심 프로세스 상에서 발생되는 대규모 트랜잭션을 처리하는 단계로 발전하며 변화하고 있다[20]. 워크플로우의 기술을 통해 날로 복잡하고 다양해지는 업무들의 흐름을 효과적으로 제어함으로써 기업 내의 비즈니스 프로세스의 성능을 극적으로 향상시킬 수 있으며, 기업의 비용 절감과 생산성의 향상을 성취할 수 있을 뿐만 아니라 고객들에게는 양질의 서비스를 제공할 수 있다. 따라서 워크플로우 기술의 도입 및 적용은 기업의 경쟁력 자체를 증가시키는 중요한 요소가 된다.

워크플로우 시스템을 조직에 도입함으로써 얻게 되는 이점은 다음과 같다.

첫째, 조직 내 업무처리의 흐름을 일관적이고 체계적으로 관리할 수 있다.

둘째, 업무 처리 절차의 자동화를 구현함으로서 사무업무 비용을 절감할 수 있다. 셋째, 조직 내의 업무처리의 신속성을 구현할 수 있을 뿐 아니라 이는 결국 고객에 대한 서비스 향상으로 이어져 조직의 매출과 생산성을 크게 증가시킬 수 있다. 넷째. 조직 내의 업무처리에 대한 자동화하고 그의 변경 및 처리 이력들을 조회 가능하게 함으로서 업무의 효율성을 극대화 할 수 있다. 다섯째, 업무 흐름을 가시화하고 그에 따라 모니터링 할 수 있기 때문에 업무의 생산성을 향상시켜준다. 여섯째, 신규 직원의 경우에도 자신의 업무를 쉽게 파악할 수 있어 업무 투입에 필요한 교육비용을 줄일 수 있으므로 전체적인 생산성을 향상시킬 수 있다. 이런 모든 이점은 치열한 경쟁 환경에서 기업의 경쟁력을 확보시켜 준다.

2.2 UML

본 절에서는 UML의 전반적인 소개와 구성에 대해 알아본 후 UML의 구성요소중의 하나인 활동 다이어그램의 개념과 구성요소 그리고 특징에 대해서 알아본다.

2.2.1 UML(Unified Modeling Language)

UML은 객체지향 시스템 모델을 작성하기 위한 객체지향적 분석과 설계 개념과 표기법을 제공한다. UML은 Rumbaugh의 OMT(Object Modeling Technology)의 방법론과 Booch의 OOAD(Object oriented Analysis and Design)방법론과 Jacobson의 OOSE(Object Oriented Software Engineering) 방법론 등의 장점을 통합하여 만든 모델링 개념의 공통집합으로, 객체지향분석 및 설계 방법론의 표준지정을 목표로 제안되었다[21].

UML은 복잡한 대규모 시스템모델링에서 성공을 거둔 소프트웨어공학의 Best of best만을 모은 것이다. 우리가 살아가는데 있어 공통적인 약속인 언어가 있듯이 모델링 하는데 있어 공통적인 약속이 필요한데 그것이 바로 UML인 것이다. UML은 Data Modeling, Business Modeling, Object Modeling, Component Modeling 등에 널리 사용되고 있다. 또한 UML은 소프트웨어 시스템 산출물의 명세화, 시각화, 구축, 문서화에 사용되는 언어로 비즈니스모델링이나 소프트웨어 시스템이 아닌 경우에도 사용될 수 있다[21]. 1997년 11월에 UML 1.1 이 OMG(Object Management Group)에 의해 표준으로 채택되었다. 2001년에 UML 1.4, 1.5 버전이 나왔으며, UML은 소프트

웨어 업계의 명실상부한 표준이 되어, 계속 수정 보완 되어, 2004년 10월에는 UML 2.0이 나온 상태이다. UML 2.0은 이전 버전인 1.5보다 훨씬 더 많은 수의 그래픽 요소들과 다양한 다이어그램을 제시하여 워크플로우 모델링시 표현의 풍부함을 제공하고 있다. OMG에서 표준객체지향 모델링 언어로 채택되었기 때문에 여러 회사들은 이 방법론을 적용하여 소프트웨어를 개발하고 있다[21]. UML은 소프트웨어 개발 과정에서 산출되는 산출물, 혹은 설계 내용들을 모델링시에 여러 다이어그램을 통해 소프트웨어 개발 시의 산출물을 비주얼하게 제공하며, 개발자들 간의 정보 전달 이해를 높이는 역할을 한다.

2.2.2 UML의 구성

UML은 크게 기호로 사용되는 사물(Thing)과 이들 간의 관계(Relationship), 사물과 관계를 기반으로 그릴 수 있는 하나의 큰 그림 즉 다이어그램(Diagrams) 이며, 이들은 관리 기준이 되는 소프트웨어 아키텍처로 구성되어 있다. 다이어그램은 Use Case Diagram, Activity Diagram, Sequence Diagram, Collaboration Diagram, Class Diagram, Component Diagram, Deployment Diagram 등으로 구성된다.

각 다이어그램을 살펴보면, 쓰임새 다이어그램(Use case Diagram)은 요구분석 과정에서 시스템과 그 외부 간의 상호 작용을 묘사하는데 사용된다. 구성요소는 시스템에게 무엇인가를 요청하거나 요청받는 Actor와 시스템의 기능을 표현한 Use Case로 구성되어 있다. 활동 다이어그램(Activity Diagram)은 요구분석 과정이나 프로세스 분석과정에서 업무의 흐름을 모델링 하거나 객체의 생명주기를 표현할 때 사용된다. 순차 다이어그램(Sequence)은 주로 분석과 설계과정에서 객체 간의 메시지 전달을 시간적 흐름에 따라 분석 할

때 주로 사용되는 다이어그램으로 객체 분석과 설계과정에서 이용되며 객체와 메시지로 구성된다. 협력 다이어그램(Collaboration Diagram)은 분석 설계 과정에서 객체와 객체가 주고받는 메시지를 중심으로 작성한 동적 다이어그램이다. 클래스 다이어그램(Class Diagram)은 시스템의 구조적인 모습을 그리는 다이어그램으로 주로 설계 단계에서 만들어지는 다이어그램이다. 클래스 다이어그램의 구성요소로는 클래스, 인터페이스, 그리고 이들 간의 관계(Relationship)가 있다. 컴포넌트 다이어그램(Component Diagram)은 구현 과정에서 소프트웨어 아키텍처가 그리는 다이어그램으로 구성요소는 콤포넌트, 인터페이스가 있다. 배치 다이어그램(Deployment Diagram)은 다이어그램은 기업 환경의 구성과 그 위에서 돌아가는 컴포넌트들 간의 관계를 그리는 다이어그램이며 주로 SE(system engineer)가 사용한다[21].

2.2.3 UML 활동 다이어그램(AD ; Activity Diagram)

UML AD(이하 UML AD는 UML 활동 다이어그램을 의미한다)란 순서도의 일종으로 발생하는 활동을 강조하여 이 다이어그램에서는 작업 과정이나 어떤 동작의 단계를 보여준다. 이러한 AD에서 기존 순서도와는 달리 어떠한 행위에 따라 객체의 상태를 표기할 수 있다는 것이다. AD는 흐름도로서 활동으로부터 활동으로 전달되는 제어 흐름과 객체 간 제어흐름을 표현한다. 이 다이어그램은 처리 단계(이것을 활동(Activity)이라고 부른다), 결정 위치, 분기 처리를 표현할 수 있어서, 업무 처리과정이나 오퍼레이션에서 처리되는 일들을 효과적으로 나타내는데 유용하게 사용된다. 이 과정은 시스템 분석 과정의 일부에 속한다. AD는 사용자에게 시작과 끝을 명확하게 보여주는 특성 때문에 쓰임새(Use Case)들 사이에서는 물론이고 쓰임새 내부에서도 작업흐름

을 잘 보여준다.

AD의 구성요소는 크게 워크플로우에서 실제 단위 업무를 나타내는 동적활동과 그 동적활동을 지원하는 정적활동의 노드를 의미하는 그래픽노드(Graphic nodes)로 주로 사용하는 노드로는 ActionNode, ActivityFinal Node, InitialNode, DecisionNode, ForkNode, JoinNode, MergeNode, SendSignalAction, AcceptEventAction 등이 있으며, 각 활동 간의 흐름이나 객체간의 흐름을 나타내는 그래픽 경로(Graphic paths)에는 ControlFlow, ObjectFlow 그리고 그 외에 기타요소로 그래픽 엘리먼트(Graphic elements)로 ActivityPartition, ExceptionHandler, ExpansionRegion 등으로 구분할 수 있다[4].

본 논문에서는 AD에서 주로 사용되는 요소인 그래픽노드와 그래픽 경로를 알아보고 그래픽 노드 중에서도 AD에 주로 사용되는 노드들과 그래픽 패스를 EMFG로 표현해 보고자 한다. 그래픽 엘리먼트는 빈번하게 사용되지 않거나 예외적인 상황에서 사용된다. 일반적인 UML 활동다이어그램을 기반으로 하는 워크플로우를 EMFG로 표현하고자 하므로 본 논문에서는 그래픽 엘리먼트는 고려하지 않는다.

AD의 주요 구성요소는 그래픽 노드로서 UML AD을 이용하여 워크플로우를 모델링시 가장 빈번히 사용되는 요소들이다. AD의 구성요소 중 주요 그래픽 노드에 대하여 살펴보자.

ActionNode는 활동 다이어그램에서 가장 많이 사용하는 노드로서 각 단위 활동을 의미하며, 표현은 둥근 모서리를 가진 직사각형으로 표현한다. ActionNode는 각 노드 간에 제어흐름이나 데이터 흐름을 들어가고 나가는 Flow를 가지며, 입력 조건이 만족할 때까지 수행되지 않는다. 아래 그림은 ActionNode를 나타낸 것이다.

그림 1. ActionNode

ActivityFinalNode은 활동다이어그램의 종료를 나타내는 노드이다. 표현은 검은 원이 내부에 있는 이중 원으로 표시한다.

그림 2. ActivityFinalNode

InitialNode는 활동 다이어그램을 시작시키는 노드로서 워크플로우의 시작을 나타내며 검은 원으로 나타낸다.

그림 3. InitialNode

DecisionNode는 결정을 나타내는 노드로서 객체나 활동의 흐름, 즉 업무의 흐름이 조건에 의하여 분기되는 경우에 사용하며, 하나의 입력에서 여러 활동 상태 중에서 조건에 의한 다른 활동으로 분기된다.

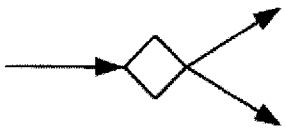


그림 4. DecisionNode

ForkNode는 하나의 활동에서 여러 다른 활동으로 전이 되는 경우이며, ForkNode는 병렬 처리를 가능하게 한다.

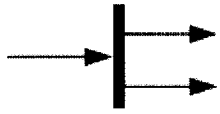


그림 5. ForkNode

JoinNode는 여러 개의 활동이 모두 만족되어 한 활동의 상태로 전이되는 경우에 사용된다. JoinNode는 병렬처리의 흐름들을 연결하여 단일의 처리흐름을 회복시킨다.

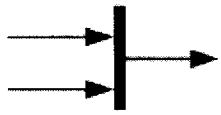


그림 6. JoinNode

MergeNode는 여러 활동의 상태 중에서 하나만 만족이 되어도 다음 상태로 전이되는 경우에 사용된다. 아래와 같이 나타낸다.

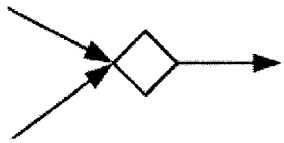


그림 7. MergeNode

ObjectNode는 각 객체를 표현시 나타내는 노드이다. 직사각형이나 둥근 모서리를 가진 원에 작은 사각형을 표시하여 나타낸다.

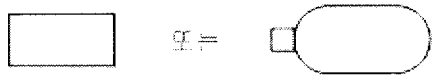


그림 8. ObjectNode

SendSignalAction는 신호를 다른 활동이나 객체에 보내는 노드이며 오른쪽이 삼각으로 튀어나온 직사각형으로 나타낸다.

그림 9. SendSignalAction

AcceptEventAction은 다른 활동이나 객체에서 신호를 받아들이거나 보내는 역할을 하는 노드이며, 왼쪽이 삼각으로 들어간 직사각형으로 나타낸다.

그림 10. AcceptEventAction

다음은 그래픽 요소들의 흐름을 연결해주는 그래픽 패스(Graphic paths)들에 대해서 알아본다.

그래픽 패스 요소들은 그래픽 노드들을 연결하여 상태의 흐름과 객체의 흐름 등을 제어한다. 그래픽 요소에는 활동의 상태를 제어하는 ControlFlow와 객체의 흐름을 제어하는 ObjectFlow가 있다. ControlFlow는 활동노드들의 앞이나 뒤에서 연결되어 흐름을 제어하며 표현은 화살표모양으로 표시한다. ObjectFlow는 객체간의 연결을 하거나 활동과 객체를 연결하며 표현은 ControlFlow와 같이 화살표로 나타낸다.

그림 11. ControlFlow

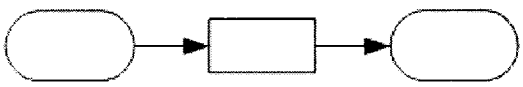


그림 12. ObjectFlow

그 외 AD뿐만 아니라 UML의 많은 다이어그램을 위한 그래픽 요소가 UML 2.0에서 더욱 많이 수정 추가되었으나 본 논문에서는 UML 1.5에서 주로 사용되었던 노드들을 기준으로 현재 UML AD에서 주로 사용되는 요소들만 고려한다.

2.3 EMFG

본 절에서는 EMFG의 개념과 정의에 대해 알아본 후 EMFG의 동작과 특징에 대해서 알아본다.

2.3.1 EMFG의 정의

동기나 비동기의 이산제어 시스템을 설계하여 구현하거나 분석하는데 적합한 EMFG는 박스(box), 트랜지션(transition), 아크(arc)들로 구성되는 마크를 갖는 방향성 선도로 정의되며, 다음과 같이 5개의 구성 원소로 이루어져 있다 [14, 19].

$$G = \{B, T, I, O, M\}$$

B : 박스(Box)의 집합

T : 트랜지션(Transition)의 집합

I : 트랜지션(Transition)에 대한 입력함수

O : 트랜지션(Transition)에 대한 출력함수

M : 각 박스(Box)의 마크 상태

박스는 상태(개념적인 상태, 제어 상태 등)를 나타내며, 상태의 만족 여부는 박스 내에 마크를 두어 표시한다. 즉, 박스의 상태가 만족되면 마크를 두고, 만족되지 않으면 마크를 두지 않는다.

트랜지션은 박스(들)의 상태가 조합되어 박스 자신의 상태가 변화하거나 다른 박스(들)의 상태를 변화시키는 곳, 즉 전이가 일어나는 곳으로 일반 트랜

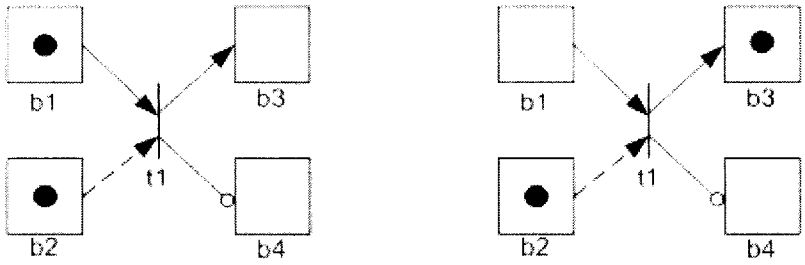
지선과 시간 트랜지션이 있다. 여기서 전이가 일어나는 과정을 트랜지션이 점화(fire)한다고 한다. 아크는 박스와 트랜지션 사이에서 트랜지션의 점화 조건을 결정하고, 트랜지션이 점화될 때 박스의 마크 상태를 결정한다. 아크의 종류에는 일반아크와 역아크, 조건아크로 구분된다.

2.3.2 EMFG의 동작

EMFG는 트랜지션의 점화조건이 만족되었을 때 동작한다. 트랜지션의 점화는 트랜지션의 모든 입력 박스(들)의 마크 상태가 모두 조건에 부합되어야 한다. 이 점화조건들은 연결된 아크의 종류에 따라 달라진다. 즉 일반/역 아크로 연결된 입력 박스에는 마크가 있/없어야 조건에 부합되고, 조건 아크로 연결된 입력 박스에는 마크가 있어야 조건에 부합된다.

트랜지션이 점화하는 경우, 트랜지션에 연결된 박스의 마크상태는 연결된 아크의 종류에 따라 다르게 변화한다. 즉, 일반 아크로 연결된 입력 박스의 마크는 소멸되고 조건 아크로 연결된 입력 박스의 마크는 그대로 유지되며, 트랜지션에서 일반/역 아크로 연결된 출력 박스에는 마크가 없/있으면 생성/소멸되고, 마크가 있/없으면 그대로 유지된다[11, 15, 19].

그림 13은 트랜지션의 점화에 따른 EMFG의 동작에 대한 예이다.



a) t1 점화 전　　b) t1 점화 후

그림 13. EMFG의 점화동작의 예

그림 13 a)에서 일반아크로 연결된 b1과 조건아크로 연결된 b2에 마크가 존재하므로 t1은 점화 가능하다. b)는 a)에서 t1이 점화한 후의 EMFG 동작을 보여준다. 점화 후에는 b2는 조건아크로 연결되어 있으므로 마크가 소멸하지 않고 b3에는 마크가 생성되며, b4는 역아크로 연결되어 있으므로 마크가 소멸된다.

2.3.3 EMFG의 특징

EMFG는 페트리넷에서 파생된 그래프 이론으로 페트리넷의 도달가능성, 생동성, 안전성 등의 모든 성질을 가지고 있다. 특히, EMFG는 페트리넷에서 시스템의 안전성을 확보하기 위해 파생된 이론이므로 시스템의 안전성을 쉽게 나타낼 수 있다. 박스가 안전하면 그 박스에 있는 마크의 수는 0이거나 1이 되며, 모든 박스가 안전하면 EMFG의 안전성, 즉 시스템의 안전성을 확보할 수 있다.

EMFG의 용도를 살펴보면 설계자의 생각을 그대로 표현하기에 적합한 강력한 설계 도구로서 시스템의 개념적 설계나 시스템 동작의 상세 설계 가능하고, EMFG를 구성하고 있는 요소를 각각 일대 일로 변환하면 직접적인 회

로를 얻을 수 있어 시스템의 구현이 쉽다.

또한 기존의 EMFG를 단순화시켜 표현하므로 시스템의 설계가 용이하고, 수학적으로 표현가능하며, 시스템의 분석이나 해석이 쉽게 이루어지고 컴퓨터 프로그램에 의해서도 그 분석 및 해석이 가능하다는 장점을 가지고 있다.

본 논문에서는 EMFG의 동작 상태를 이용하여 UML AD를 표현하는 방법을 제시하고 EMFG로 표현된 UML AD에 대하여 수행완료 가능성을 검증하여 워크플로우의 오류 여부를 판단하고, UML AD의 동작 흐름에 대해서 알아보고자 한다.

Ⅲ. UML 활동 다이어그램의 EMFG 표현

본 장에서는 UML AD(Activity Diagram; 활동 다이어그램)를 EMFG로 표현하는 방법에 대하여 논하기로 한다. 그리고 본 연구에서는 UML AD의 노드들 중에서 필수적이고 가장 많이 사용되는 8개 노드(ActionNode, ObjectNode, ActivityFinalNode, InitialNode, DecisionNode, MergeNode, ForkNode, JoinNode) 및 플로우(flow)에 대해서만 고려하기로 한다.

3.1 용어 정의

본 연구에서 취급하는 UML AD의 노드들 중 4개는 상태를 의미하고, 4개는 제어의 의미를 나타내므로 다음과 같이 정의한다.

정의 1) UML AD의 노드들 중에서 상태를 의미하는 ActionNode, ObjectNode, ActivityFinalNode 및 InitialNode를 상태 노드(state node)라 하고, 제어의 의미를 나타내는 DecisionNode, MergeNode, ForkNode 및 JoinNode를 제어 노드(control node)라 한다.□

상태 노드는 임의의 시간에서 어떤 상태에 있게 되는데, 즉 상태 노드의 상태가 만족되든지 아니면 만족되지 않은 상태에 있게 된다. 그리고 상태 노드의 상태는 종류에 따라 다음과 같이 다른 의미를 가진다.

· ActionNode의 상태가 만족된다는 것은 ActionNode가 활동 중임을 의미하고, 만족되지 않는다는 것은 비활동 중임을 뜻한다.

· ObjectNode의 상태가 만족된다는 것은 Object가 생성되었음을 의미하고, 만족되지 않는다는 것은 Object가 생성되지 않았거나 사용되지 않음을 뜻한다.

· ActivityFinalNode의 상태가 만족된다는 것은 활동의 종료를 의미하고, 만족되지 않는다는 것은 활동이 종료되지 않았음을 뜻한다.

· InitialNode의 상태가 만족된다는 것은 활동의 시작을 의미하고, 만족되지 않는다는 것은 활동의 시작이 아님을 뜻한다.

그리고 UML AD의 각 노드들은 상호간에 플로우(flow)로 연결되는데, 이러한 플로우에는 노드로 입력되는 것과 노드에서 출력되는 것이 있을 수 있으므로 다음과 같이 정의한다.

정의 2) 노드로 입력되는 플로우를 그 노드에 대한 입력 플로우(input flow)라 하고, 노드에서 출력되는 플로우를 그 노드에 대한 출력 플로우(output flow)라 한다.□

UML AD 구조는 정의1에서 정의된 노드들과 정의2에서 정의된 플로우들로 구성되고, 이 구조를 상태의 흐름에 따라 구분하면 3가지 구조가 있을 수 있으므로 다음과 같이 정의한다.

정의 3) UML AD에서 상태 노드와 상태 노드가 플로우로 결합된 구조를 SS(State-State) 구조, 상태 노드와 상태 노드 사이에 제어 노드가 플로우로 결합된 구조를 SCS(State-Control-State) 구조, 상태 노드와 상태 노드 사이에 2개 이상의 제어노드들이 플로우로 결합된 구조를 SMCS (State-MultiControl-State) 구조라 한다.□

SS 구조는 하나의 입력 노드와 하나의 출력 노드가 플로우로 연결된 구조이다. SCS 구조는 하나 이상의 입력 노드와 하나 이상의 출력 노드가 하나의 제어 노드를 사이에 두고 플로우들로 연결된 구조이며 제어 노드의 종류에 따라 입력 노드와 출력 노드의 수가 달라진다. SMCS 구조는 하나 이상의 입력 노드와 하나 이상의 출력 노드가 두 개 이상의 제어 노드를 사이에 두고 플로우들로 연결된 구조이며 제어 노드의 종류에 따라 입력 노드와 출력 노드의 수 및 내부 연결 구조가 달라진다.

UML AD의 각 구조에 대한 예를 그림 14에 나타내었다.

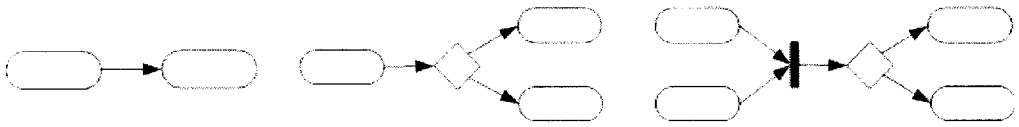


그림 14. UML AD에 나타나는 구조의 예

SS 구조는 어떤 상태에서 다른 상태로 변화됨을 의미하는 구조이고, SCS 구조와 SMCS 구조는 어떤 상태에서 제어되어 다른 상태로 변화됨을 의미한다. 그리고 각 구조에서 앞쪽에 오는 상태 노드는 ActionNode, ObjectNode 및 InitialNode 등이 될 수 있으며, 뒤쪽에 오는 상태 노드는 ActionNode, ObjectNode 및 ActivityFinalNode 등이 될 수 있다.

3.2 UML AD의 EMFG 표현

UML AD에서는 각 노드의 상태가 변화하여 이동함으로써 동작한다. 이러한 상태는 상태 노드의 상태로 나타나게 되고, 제어 노드의 동작에 따라 상태가 변화하여 이동하게 되는 셈이다. 따라서 UML AD를 EMFG로 변환할 때는 상태와 제어 두 가지 관점에서 고려되어야 한다.

우선 상태 노드의 상태를 EMFG의 박스 상태와 대응시킨다면, 다음과 같이 상태 노드를 EMFG로 표현할 수 있다.

규칙 1. UML AD에서 상태 노드의 EMFG 표현 : UML AD에서 상태 노드는 EMFG에서 박스로 표현하고, 노드의 상태에 대한 만족 여부는 해당 박스의 마크 유무로 표현한다. 이때 상태 노드명은 박스명과 함께(또는 별도) 표기한다.□

예를 들어 그림 15a)의 ActionNode인 'Send Payment' 노드를 그림 15b)또는 그림 15c)와 같이 EMFG의 박스 b1로 표현할 수 있으며, 'Send Payment' 노드가 활동 중일 때는 b1에 마크가 존재하게 된다.

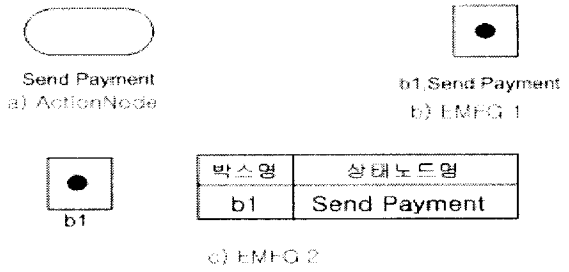


그림 15. 상태 노드의 EMFG 표현 예

특히 InitialNode와 ActivityFinalNode는 UML AD의 시작과 끝에 나타나므로 이에 대응되는 EMFG 박스를 시작박스(s; start box)와 ·(e; end box)라

한다.

다음은 UML AD의 구조를 EMFG로 표현하는 방법에 대하여 알아보기로 한다.

SS 구조는 어떠한 제어의 간섭도 받지 않고 상태 변화만 일어나는 구조이므로 다음과 같이 EMFG로 표현될 수 있다.

규칙 2. UML AD에서 SS 구조(S_S)의 EMFG 표현 : S_S의 입력 노드를 I, 출력 노드를 O라 할 때, S_S는 다음 순서에 따라 EMFG로 표현한다.

1) I와 O를 각각 규칙1에 따라 박스로 표현한다.

2) I와 대응되는 박스를 입력으로 가지고 O와 대응되는 박스를 출력으로 가지는 하나의 트랜지션을 구성한다.□

예를 들어 그림 16a)의 ActionNode와 ObjectNode로 구성되는 SS 구조를 EMFG로 표현하면 그림 16b)와 같다. 그림 16a)에서 SS 구조의 상태 흐름은 그림 16b)에서 EMFG의 마크 이동으로 파악된다.

a) SS 구조 b) a)의 EMFG

그림 16. SS 구조의 EMFG 표현

SCS 구조는 포함된 제어 노드의 제어에 의해 상태 변화가 일어나므로 제어 노드의 제어 방법에 따라 다르게 상태를 변화시킨다. 따라서 제어 노드의 종류에 따라 EMFG로 표현하는 방법도 서로 다르게 된다.

DecisionNode는 내부에 두 개 이상의 조건을 가지고, 각 조건의 만족 상태에 따라 다르게 상태가 변화되므로 DecisionNode를 가지는 SCS 구조는 다음과 같이 EMFG로 표현될 수 있다.

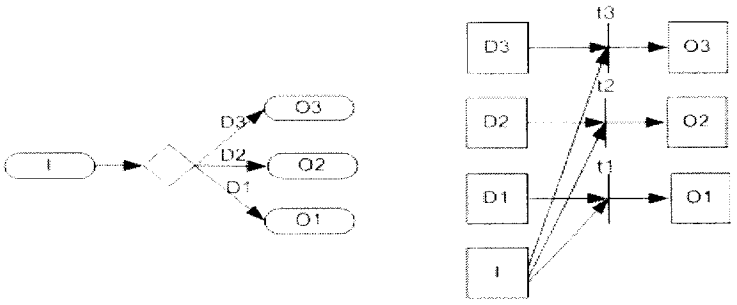
규칙 3. UML AD에서 DecisionNode 제어 노드를 가진 SCS 구조(S_{CD})의 EMFG 표현 : S_{CD}의 DecisionNode 제어 노드가 n개의 조건을 가진 경우, S_{CD}에서 입력 노드를 I, DecisionNode 제어 노드의 각 조건을 D_i, D_i에 연결된 출력 노드를 O_i라 할 때, S_{CD}는 다음 순서에 따라 EMFG로 표현한다. 여기서 i는 1 이상이고 n 이하인 정수이다.

1) I 및 각 O_i를 규칙1에 따라 각각 해당 박스로 표현한다.

2) D_i를 각각 하나의 박스로 표현한다. 여기서 각 박스를 조건박스라 하며, 조건 D_i의 만족여부는 대응되는 조건박스의 마크 존재유무를 의미한다.

3) I에 대응되는 박스와 D_i에 대응되는 박스를 입력으로 가지고, O_i에 대응되는 박스를 출력으로 가지는 트랜지션 t_i를 각각 구성한다. 여기서 각 박스들은 t_i에 일반아크로 연결된다.□

예를 들어 그림 17a)의 DecisionNode를 가진 SCS 구조를 EMFG로 표현하면 그림 17b)와 같다. 그림 17에서 보는 바와 같이 UML AD의 조건에 따라 상태 변화가 일어나는 모습을 EMFG에서도 쉽게 파악할 수 있다.



a) DecisionNode를 가진 SCS 구조 b) a)의 EMFG

그림 17. DecisionNode를 가진 SCS구조의 EMFG 표현

MergeNode는 입력 플로우에 연결된 입력 상태(들)가 만족되어 출력 플로우에 연결된 상태를 변화시켜 만족시키게 되므로 MergeNode를 가진 SCS 구조는 다음과 같이 EMFG로 표현될 수 있다.

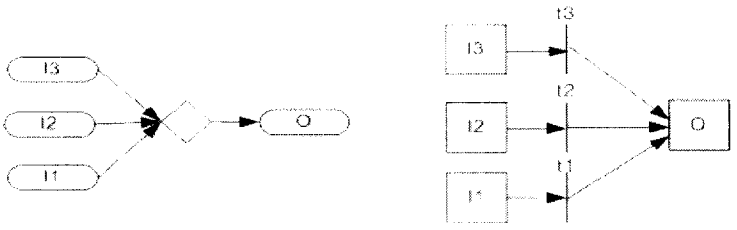
규칙 4. UML AD에서 MergeNode 제어 노드를 가진 SCS 구조(S_{CM})의 EMFG 표현 : S_{CM}의 MergeNode 제어 노드가 n개의 입력 노드를 가지는 경우, S_{CM}에서 각 입력 노드를 I_i, 출력 노드를 O라 할 때, S_{CM}은 다음 순서에 따라 EMFG로 표현한다. 여기서 i는 1 이상이고 n 이하인 정수이다.

1) I_i 및 O를 규칙1에 따라 각각 해당 박스로 변환한다.

2) I_i에 대응되는 박스를 입력으로, O에 대응되는 박스를 출력으로 가지는 트랜지션 t_i를 구성한다. 여기서 각 박스들은 트랜지션 t_i에 일반아크로 연결된다.□

예를 들어 그림 18a)의 MergeNode를 가진 SCS 구조를 EMFG로 표현하면 그림 18b)와 같다. 그림 18에서 보는 바와 같이 UML AD에서 여러 상태 중에서 하나 이상이 만족되면 다음 하나의 상태로 변하게 되는 모습을 EMFG

에서도 그대로 쉽게 파악할 수 있다.



a) MergeNode를 가진 SCS 구조 b) a)의 EMFG

그림 18. MergeNode를 가진 SCS 구조의 EMFG 표현

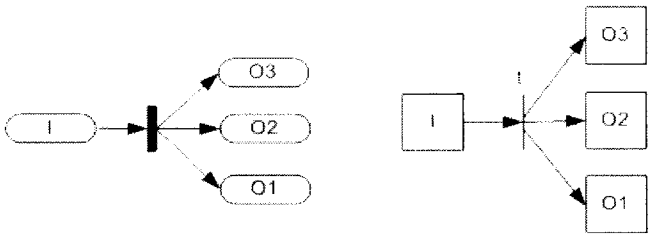
ForkNode는 입력 플로우로 연결된 상태가 출력 플로우에 연결된 상태들을 변화시키게 되므로 ForkNode를 가진 SCS 구조는 다음과 같이 EMFG로 표현될 수 있다.

규칙 5. UML AD에서 ForkNode 제어 노드를 가진 SCS 구조(S_{CF})의 EMFG 표현 : S_{CF}의 ForkNode 제어 노드가 n개의 출력 노드를 가지는 경우, S_{CF}에서 입력 노드를 I, 각 출력 노드를 O_i라 할 때, S_{CF}는 다음 순서에 따라 EMFG로 표현한다. 여기서 i는 1 이상이고 n 이하인 정수이다.

1) I 및 O_i를 규칙1에 따라 각각 해당 박스로 변환한다.

2) I에 대응되는 박스를 입력으로, 출력 노드에 대응되는 모든 박스를 출력으로 가지는 트랜지션 t를 구성한다. 여기서 각 박스들은 t에 일반아크로 연결된다.□

예를 들어 그림 19a)의 ForkNode를 가진 SCS 구조를 EMFG로 표현하면 그림 19b)와 같다. 그림 19에서 보는 바와 같이 UML AD에서 하나의 상태에서 두개 이상의 상태로 변하게 되는 모습이 EMFG에서도 파악될 수 있다.



a) ForkNode를 가진 SCS 구조 b) a)의 EMFG

그림 19. ForkNode를 가진 SCS 구조의 EMFG 표현

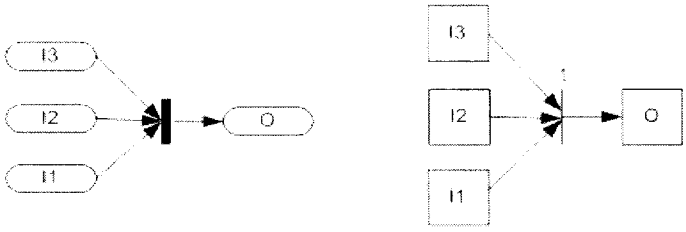
JoinNode는 두개 이상의 상태가 모두 만족될 때 다음 상태를 변화시키게 되므로 다음과 같이 EMFG로 표현될 수 있다.

규칙 6. UML AD에서 JoinNode 제어 노드를 가진 SCS 구조(S_{CJ})의 EMFG 표현 : S_{CJ}의 JoinNode 제어 노드가 n개의 입력 노드를 가지는 경우, S_{CJ}에서 각 입력 노드를 I_i, 출력 노드를 O라 할 때, S_{CJ}는 다음 순서에 따라 EMFG로 표현한다. 여기서 i는 1 이상이고 n 이하인 정수이다.

1) I_i 및 O를 규칙1에 따라 각각 해당 박스로 변환한다.

2) 입력 노드에 대응되는 모든 박스를 입력으로, O에 대응되는 박스를 출력으로 가지는 트랜지션 t를 구성한다. 여기서 각 박스들은 트랜지션 t에 일반 아크로 연결된다.□

예를 들어 그림 20a)의 JoinNode를 가진 SCS 구조를 EMFG로 표현하면 그림 20b)와 같다. 그림 20에서 보는 바와 같이 UML AD에서 두개 이상의 상태가 모두 만족하게 될 때 다음 상태로 변하게 되는 모습을 EMFG에서도 그대로 파악할 수 있다.



a) JoinNode를 가진 SCS 구조 b) a)의 EMFG

그림 20. JoinNode를 가진 SCS 구조의 EMFG 표현

SMCS 구조는 상태 노드와 상태 노드 사이에 여러 개의 제어 노드들이 있어 두 개 이상의 제어가 연속적으로 있게 된다. UML AD의 목적은 상태 변화에 초점을 두고 있으므로 상태 변화 관점에서 보면 SMCS 구조를 다음과 같이 SCS 구조로 바꾸어도 UML AD의 동작에는 영향을 받지 않는다.

정리 1) UML AD에서 상태 노드 N1, 두 개의 제어 노드 C1과 C2, 상태 노드 N2가 직렬로 구성되는 SMCS 구조 S_{MC}가 있다. S_{MC}에서 C1과 C2 사이에 ActionNode인 가상노드(pseudo Node) N_P를 두어 C1과 C2를 플로우로 연결하였을 때, N1과 C1 및 N_P로 구성되는 SCS 구조를 S_{C1}, N_P와 C2 및 N2로 구성되는 SCS 구조를 S_{C2}라 하자. 이때 S_{MC}의 동작은 S_{C1}과 S_{C2}로 구성된 구조의 동작과 서로 동일하다.

증명) 구조 S_{MC}에서 C1의 상태는 C2의 입력으로 직접 사용되어 C2의 상태를 변화시킨다. 그리고 S_{C1}의 C1의 상태는 그대로 N_P에 전달되고, 이 N_P의 상태는 S_{C2}의 N_P의 상태와 동일하므로 N_P의 상태가 그대로 C2에 입력되어 C2의 상태를 변화시킨다. 이 동작은 S_{MC}의 동작과 동일하므로 정리는 타당하다.□

보조정리 1 1) UML AD에서 상태 노드 N1, n개의 제어 노드가 있을 때 각 제어 노드를 C_i(1<= i <= n), 상태 노드 N2가 직렬로 구성되는 SMCS 구조 S_{MC}가 있다. S_{MC}에서 C_1과 C_2 사이에 가상 노드 N_{P1}, C_2와 C_3 사이에 가상 노드 N_{P2}, ..., C_n와 N2 사이에 가상 노드 N_{Pn}를 두어 각 가상 노드를 인접하는 제어 노드들과 플로우로 연결하였을 때, N1과 C_1 및 N_{P1}로 구성되는 SCS 구조를 S_{C1}, N_{P1}과 C_2 및 N_{P2}로 구성되는 SCS 구조를 S_{C2}, ..., C_n과 N_{Pn} 및 N2로 구성되는 SCS 구조를 S_{Cn}라 하자. 이때 S_{MC}의 동작은 S_{C1}, S_{C2}, ..., S_{Cn}로 구성된 구조의 동작과 동일하다.

증명) 정리 1과 같은 방법으로 증명된다. 따라서 정리는 타당하다.□

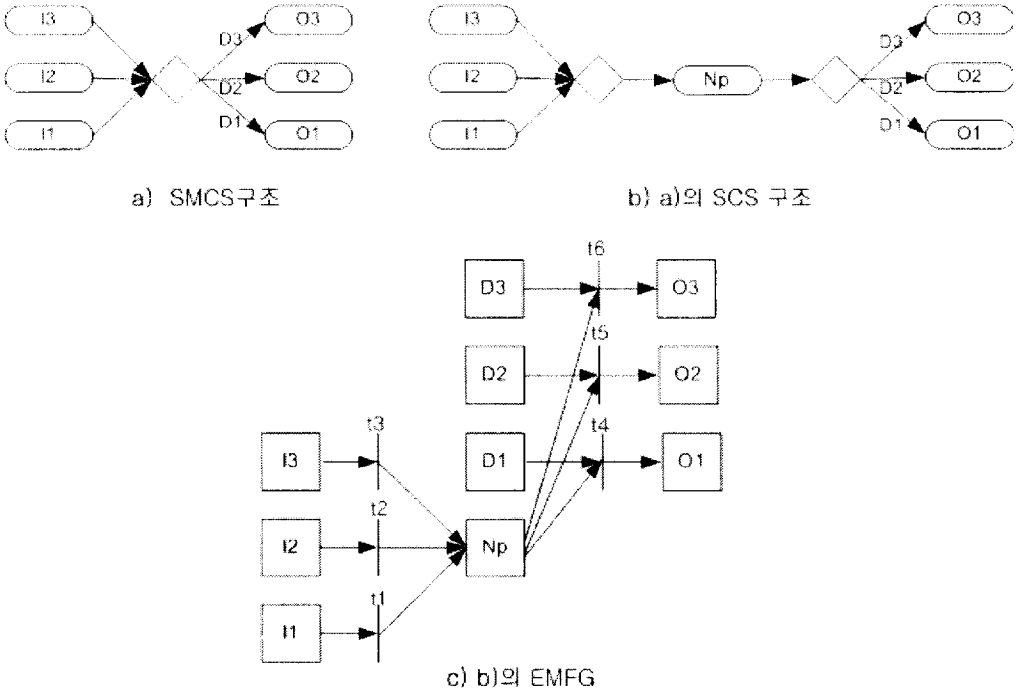
SMCS 구조는 각 제어노드들에 의해서 상태가 변하게 되므로 다음과 같이 EMFG로 표현될 수 있다.

규칙 7. UML AD에서 SMCS 구조(S_{MC})의 EMFG 표현 : S_{MC}는 다음 순서에 따라 EMFG로 표현한다.

1) S_{MC}의 동작과 동일하도록 S_{MC}를 가상노드들이 포함된 여러 개의 SCS 구조로 변환한다. 여기서의 변환은 보조정리 1-1이 적용되도록 변환되어야 한다.

2) 변환된 각 SCS 구조를 제어 노드의 종류에 따라 규칙 3, 4, 5, 6에 의해 EMFG로 표현한다.□

예를 들어 그림 21a)는 MergeNode와 DecisionNode가 연결되어 함께 표현되는 SMCS 구조이며, 규칙7에 따라 그림 21a)를 그림 21b)와 같이 가상노드 Np를 가진 SCS 구조로 변환한 후 그림 21c)와 같이 EMFG로 표현할 수 있다.



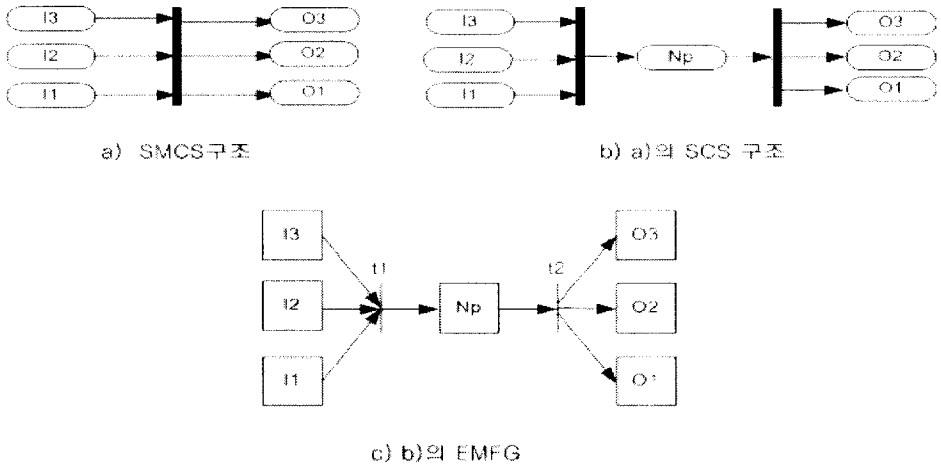
a) SMCS구조

b) a)의 SCS 구조

c) b)의 EMFG

그림 21. SMCS 구조의 EMFG 표현 예1

그림 22a)는 JoinNode와 ForkNode가 연결되어 함께 표현되는 SMCS 구조이며, 규칙7에 따라 그림 22a)를 그림 22b)와 같이 가상노드 Np를 가진 SCS 구조로 변환한 후 그림 22c)와 같이 EMFG로 표현할 수 있다.

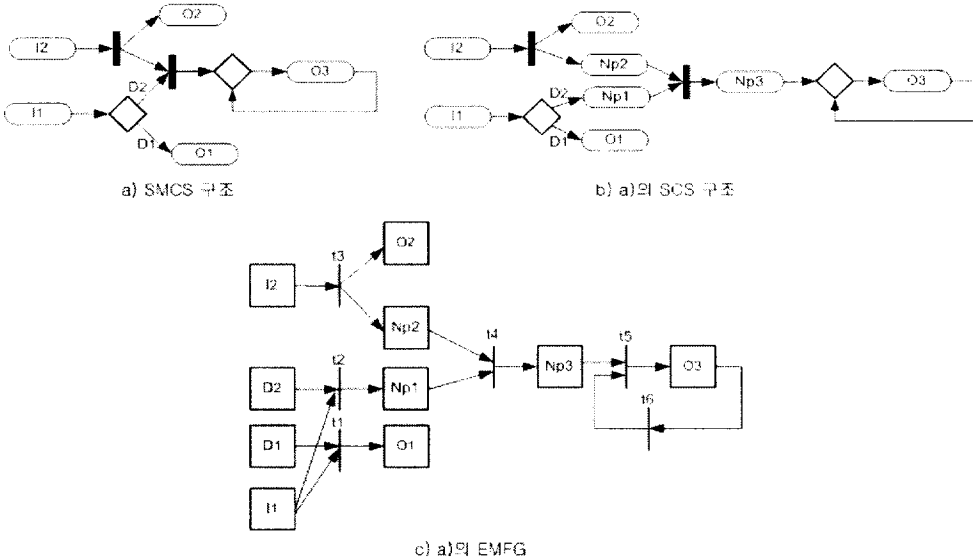


a) SMCS구조 b) a)의 SCS 구조

c) b)의 EMFG

그림 22. SMCS 구조의 EMFG 표현 예2

그림 23a)는 여러 종류의 제어노드가 연결되어 표현되는 SMCS 구조이며, 규칙7에 따라 그림 23a)를 그림 23b)와 같이 가상노드 Np를 가진 SCS구조로 변환한 후 그림 23c)와 같이 EMFG로 표현할 수 있다.



a) SMCS 구조 b) a)의 SCS 구조

c) a)의 EMFG

그림 23. SMCS 구조의 EMFG 표현 예3

3.3 UML AD의 EMFG 표현 예

본 절에서는 3.2절의 규칙들에 의해 UML 활동 다이어그램을 EMFG로 표현해 보고자 한다.

시나리오 1과 같은 고객 불만 처리 워크플로우를 UML AD로 나타내고, 이를 EMFG로 표현한다.

시나리오 1. 고객 불만 처리 워크플로우

① 고객으로부터 불만 사항이 발생(s; start)하면 이를 등록(R; register)하고, 고객에게 질문서를 보냄(SF; send form)과 동시에 불만 사항을 평가(EV; evaluate)한다.

② 고객에게 발송된 질문서는 한 달 내에 회수(조건 D1)되어 분석(AF; analyze form)되고, 분석된 결과는 불만 사항 처리에 반영된다. 만약 한 달 내에 회수되지 않으면(조건 D2) 타임아웃 처리(TO; time out)하여 분석하지 않고 ⑥의 기록 정리 단계로 간다.

③ 불만 사항 평가 결과, 경미한 경우(조건 D3)에는 처리 없이 기록 정리 후 종료하고, 경미하지 않은 경우(조건 D4)에는 불만 사항에 대하여 처리(PC; process complaint)한다.

④ 불만 사항은 질문서 분석 결과와 불만 사항 평가 결과를 조합하여 처리하고, 진행 사항을 체크(CP; check process)한다.

⑤ 진행 사항이 완전(조건 D5)하면 기록 정리 단계로 가고, 불완전(조건 D6)하면 다시 불만 사항을 처리하게 된다.

⑥ 질문서가 회수되지 않았거나 질문서 분석이 종료되었고, 불만 사항 평가

결과가 경미하거나 불만 사항이 완전히 처리된 경우는 기록 정리(AR; archive)하고, 기록 정리가 완료되면 불만 사항 처리를 종료(e; end)한다.

시나리오 1을 UML AD로 표현하면 그림 24[9]와 같다.

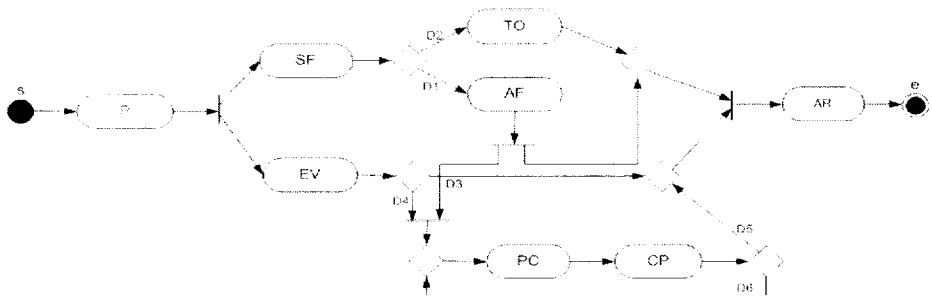


그림 24. 시나리오 1의 UML AD

규칙1에서 규칙7까지의 규칙을 사용하여 그림 24의 UML AD를 EMFG로 표현하면 그림 25와 같다. 여기서 UML AD의 노드들과 조건들의 명칭은 EMFG 표현에서 그대로 사용하였으며, Np1에서 Np9는 UML AD의 SMCS 구조에서 SCS 구조로 변환되면서 생성된 가상 노드이다.

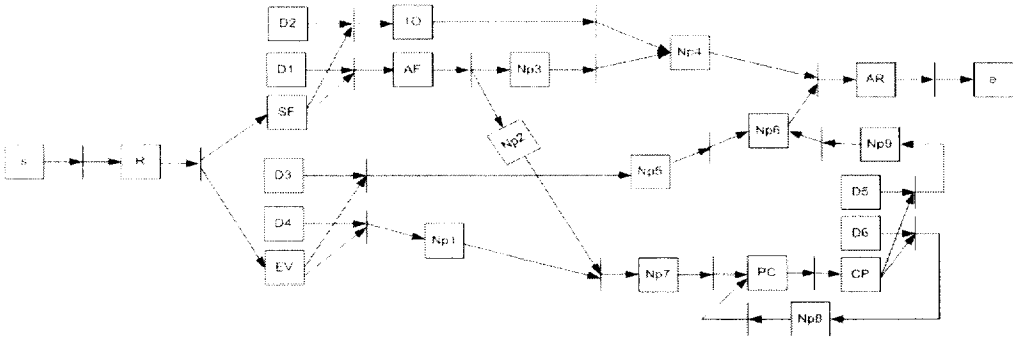


그림 25. 그림 24의 EMFG 표현

시나리오 2는 OMG(2004)의 주문처리 워크플로우를 UML AD[4]로 나타내기 위한 시나리오이다.

시나리오 2. 주문처리 워크플로우 시나리오

① 주문처리가 시작(s; start)되면 고객으로부터 주문을 받는다(RO; receive order).

② 수량 부족 등의 이유로 인하여 주문을 거절(D1; order rejected)하면 주문 처리를 종료(CO; close order)한다.

③ 주문을 수락(D2; order accepted)하면 주문내역서를 작성(FO; fill order)한다.

④ 주문내역서가 작성되면 주문 내역을 선적(SO; ship order)함과 동시에 주문 내역이 처리되었음을 확인(AA; auto approve)한다.

⑤ 주문 내역의 처리가 확인되면 송장(I; invoice)에 의해서 대금 지불을 요청(MP; make payment)하고, 이 요청이 수락(AP; accept payment)되면 주문 처리 종료 단계로 간다.

⑥ 주문 내역이 선적되고 대금 지불이 수락된 경우에 주문 처리가 완료(CO; close order)되어 워크플로우는 종료(e; end)한다.

시나리오 2를 UML AD로 나타내면 그림 26과 같다.

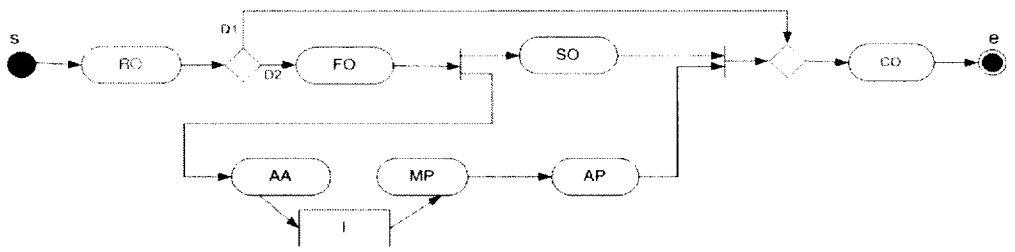


그림 26. 시나리오 2의 UML AD

그림 26의 UML AD를 EMFG로 표현하면 그림 27과 같다.

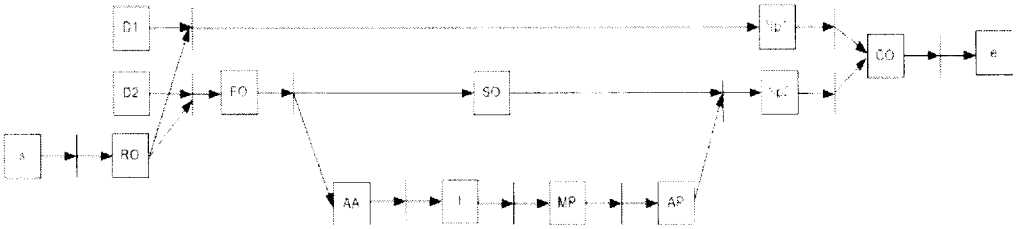


그림 27. 그림 26의 EMFG 표현

두 예에서 보는 바와 같이 UML AD는 쉽게 EMFG로 표현되며, 그 동작도 대응되는 UML AD의 동작과 동일함을 직관적으로 파악할 수 있다.

Ⅳ. UML 활동다이어그램을 표현한 EMFG의 분석

본 장에서는 UML AD를 표현한 EMFG에 대하여 그 동작을 시뮬레이션하고, 시뮬레이션 결과로써 대응되는 UML AD가 올바르게 동작되는가를 파악하고자 한다.

4.1 UML AD의 수행완료 가능성

워크플로우의 동작을 UML AD를 사용하여 모델링하는 경우, 모델링된 UML AD로써 워크플로우의 동작을 쉽게 파악할 수 있다. 이때 워크플로우의 동작 조건들은 UML AD에서 DecisionNode로 표현되고, DecisionNode의 각 조건에 따라 UML AD는 서로 다르게 수행된다. 즉 워크플로우의 각 동작 조건에 따라 워크플로우가 다르게 수행될 것이다.

그런데 어떤 조건이 부여된 상태에서 UML AD가 수행을 시작하여 종료하려면, InitialNode의 활동 상태에서 시작하여 조건이 고려되어 상태(들)가 변화하고 마지막으로 ActivityFinalNode가 활동 상태가 되어야 한다. 따라서 이와 같은 UML AD의 수행 완료 동작을 다음과 같이 정의한다.

정의 4) 조건 D에서 UML AD의 동작이 시작되어 종료될 수 있으면 'UML AD는 조건 D에서 수행 완료 가능하다'라고 하고, 고려 가능한 모든 조건들에 대하여 UML AD가 수행 완료 가능하면 'UML AD는 수행 완료 가능하다'라고 한다.□

만약 어떤 조건 D에서 UML AD가 수행완료가능하지 않으면, 조건 D에서 수행을 시작하여 종료하지 못할 것이므로 이에 대응되는 워크플로우의 동작도 제대로 완료되지 않을 것이다. 이는 UML AD가 잘못 모델링되었음을 의미한다. 따라서 워크플로우를 UML AD로 모델링할 경우는 논리적으로 합당하게 모델링해야 할 뿐만 아니라 고려 가능한 모든 조건에서 수행 완료 가능하도록 모델링해야 한다.

그런데 UML AD의 수행 완료 가능성을 검사하기 위해서는 고려 가능한 모든 조건에서 수행 완료 가능한지 수작업으로 검사해야 하는데, 이 작업은 규모가 작은 워크플로우인 경우는 크게 문제가 되지 않지만 규모가 큰 워크플로우인 경우는 쉽지 않은 문제일 수 있다.

따라서 본 연구에서는 UML AD를 EMFG로 표현하고 이를 동적 시뮬레이션하여 모델링된 UML AD가 수행 완료 가능한지 검사하려고 한다.

4.2 UML AD를 표현한 EMFG의 초기 마크 벡터

UML AD의 동작은 이를 표현한 EMFG의 동작과 동일하므로 UML AD의 수행 완료 가능성을 검사하려는 경우 대응되는 EMFG가 수행 완료 가능한지 검사하면 된다.

EMFG를 동작시키기 위해서는 초기 마킹을 부여해야 한다. 그런데 UML AD를 표현한 EMFG의 초기 마킹은 다음 조건을 만족해야 한다.

정리 2) UML AD를 표현한 EMFG의 초기 마킹은 트랜지션의 입력으로만 작용하는 박스(들)에는 마크가 존재할 수 있고 그 외의 박스에는 마크가 존재하지 않도록 구성되어야 한다.

증명) UML AD가 수행되려면 InitialNode가 활동 상태로 되어야 하고 내부의 각 DecisionNode의 조건이 결정되어 있어야 하며 이외의 어떠한 노드도 활동 상태에 있어서는 안된다. 이는 시작 박스에 마크를 두어야 하고 각 조건 박스의 마킹은 해당되는 DecisionNode의 조건 만족 여부에 따라 결정되어야 하며 이외의 박스들에는 마크가 존재하지 않음을 의미한다. EMFG에서 시작 박스와 조건 박스(들)은 트랜지션의 입력으로만 작용하고 이외의 모든 박스들은 트랜지션의 출력 박스로도 작용한다. 따라서 정리는 타당하다.□

예를 들어 그림 24의 EMFG에서 트랜지션에 대하여 입력으로만 작용하는 박스는 시작 박스 s 및 조건 박스들(D1, D2, D3, D4, D5 및 D6)이다. 따라서 이 EMFG의 초기 마킹은 s에는 마크를 두고 각 조건 박스들은 선택적으로 마크를 두며 이외 다른 박스들은 마크를 두지 않아야 한다.

그리고 UML AD에서 하나의 DecisionNode가 n 개의 조건들로 구성되는 경우, 이를 표현한 EMFG의 조건 박스들의 수는 n 개이고, 이 DecisionNode

의 조건들의 만족 상태는 조건 박스들의 마킹 상태로 표현된다. 이때 이 조건 박스들의 마킹을 조건 마킹(Decision Marking)이라 한다. 물론 UML AD에서 여러 개의 DecisionNode가 존재하는 경우에도 각 DecisionNode마다 마찬가지로 설명될 수 있다. 이 경우 EMFG의 전체 조건 박스들의 마킹을 전체 조건 마킹이라 한다.

EMFG의 초기 마킹을 결정할 때, UML AD에서 DecisionNode를 표현한 EMFG의 조건 박스의 마킹 상태는 다음 조건을 만족해야 한다.

정리 3) EMFG의 초기 마킹을 결정할 때, UML AD에서 하나의 DecisionNode를 표현한 EMFG의 조건 박스들은 오직 하나의 조건 박스에만 마크가 존재할 수 있다.

증명) UML AD를 수행시키기 전에 각 DecisionNode의 상태를 결정할 수 있다. 이때 하나의 DecisionNode는 여러 조건들로 구성되지만 만족되는 조건은 오직 하나뿐이다. 이는 대응되는 EMFG의 조건 박스들 중 하나의 박스에만 마크가 존재함을 뜻하므로 정리는 타당하다.□

보조정리 3-1) UML AD에서 n 개의 조건을 가진 하나의 DecisionNode를 표현한 EMFG의 조건 마킹의 수는 n 개이다.

증명) 정리 3에 의하여 n 개의 조건은 서로 배타적이므로 정리는 타당하다.□

예를 들어 그림 24에서 조건 D1과 D2를 가진 DecisionNode는 UML AD가 수행될 때 조건 D1이 만족되든지 아니면 조건 D2가 만족되어야 한다. 이는 그림 25에서 박스 D1과 D2 중에서 하나의 박스에만 마크가 존재해야 함을 의미한다. 즉 조건 마킹은 (1, 0) 또는 (0, 1)이 되어야 하고, 조건 마킹의 수

는 DecisionNode에서의 조건 수와 동일한 2 개이다.

또한 UML AD에 다수의 DecisionNode가 있을 때, 이를 표현한 EMFG의 전체 조건 마킹은 다음과 같이 구할 수 있다.

정리 4) m 개의 DecisionNode를 가지는 UML AD를 EMFG로 표현한 경우, m 개의 DecisionNode에 대하여 각각 이를 표현한 조건 박스들의 조건 마킹을 구하여 이들을 조합하면 EMFG의 전체 조건 마킹이 된다.

증명) UML AD가 수행되려면 모든 DecisionNode마다 하나씩의 조건이 결정되어야 한다. 이는 각 DecisionNode마다 대응되는 조건 박스들의 조건 마킹이 결정되어야 하고 EMFG의 전체 조건 마킹은 모든 노드에 대한 조건 마킹들로 구성되어야 함을 의미하므로 정리는 타당하다.□

보조정리 4-1) m 개의 DecisionNode를 가지는 UML AD에서 i 번째 DecisionNode를 D_i, D_i를 구성하는 조건의 수를 N_i라 할 때, 이와 같은 UML AD를 표현한 EMFG의 전체 조건 마킹 수 S는 식 1과 같다.

$$S = \prod_{i=1}^{m} Ni \qquad \text{(식 1)}$$

증명) D_i를 표현한 EMFG의 조건 박스들의 조건 마킹 수는 보조정리 3-1에 따라 N_i 개이고, 이는 각 DecisionNode를 표현한 EMFG의 조건 박스들에도 동일하게 적용되므로 EMFG에서 전체 조건 마킹 수 S는 식1과 같다. 따라서 정리는 타당하다.□

예를 들어 그림 24에서 DecisionNode는 3 개로서 각 노드는 (D1, D2),

(D3, D4), (D5, D6) 조건들을 가진다. 이를 표현한 그림 25의 EMFG에서 각 노드에 해당하는 조건은 2 개씩이므로 각 노드의 조건 마킹은 (1, 0)와 (0, 1)이 되고, EMFG의 전체 조건 마킹은 세 노드의 조건 마킹의 조합(D1, D2, D3, D4, D5, D6)으로 구성되므로 (1, 0, 1, 0, 1, 0), (1, 0, 1, 0, 0, 1), ..., (0, 1, 0, 1, 0, 1) 등 8(= 2 * 2 * 2) 개가 있게 된다.

UML AD의 수행 완료 가능을 검사하려면 이를 표현한 EMFG를 동작시켜 제대로 수행되는지를 검사하여야 하는데, EMFG를 동작시키기 위해서는 먼저 EMFG에 초기 마킹을 하고 동작시켜야 한다. EMFG의 초기 마이킹은 UML AD의 수행 동작 초기 상태를 나타내므로 DecisionNode들의 조건들이 결정된 상태에서 InitialNode를 활성화시키는 동작을 의미한다.

따라서 EMFG의 초기 마킹은 시작 박스(s), 조건 박스들 및 이외의 박스들의 마크 상태로 결정되는데, 조건 박스들의 마크 상태는 정리 4에 이하여 EMFG의 전체 조건 마킹으로 결정할 수 있고, EMFG가 동작을 시작하려면 시작 박스에 마크를 두어야 하며, 정리 2에 의하여 그 외의 박스들에는 마크를 두어서는 안된다. 이때 EMFG의 초기 마킹을 벡터로 표현한 것을 EMFG의 초기 마크 벡터라 한다. EMFG의 초기 마크 벡터를 구성할 때는 박스 순서와 관계없이 마크 순서를 정할 수 있고, 다음과 같이 구할 수 있다.

정리 5) EMFG의 마크 벡터 M을 (시작 박스, 조건 박스들, 나머지 박스들, 종료 박스) 순의 마크 상태로 표현할 때, 초기 마크 벡터 M_0는 식 2와 같다.

$$M_0 = (1, \text{전체 조건 마킹 상태}, 0, 0, ..., 0) \qquad \text{(식 2)}$$

증명) M_0에서 시작 박스(s)의 마크 존재는 당연하고, 조건 박스들의 마크 상태는 전체 조건 마킹 상태이며, 나머지 박스들 및 종료 박스의 마크 상태는 정리 2에 의하여 0이므로 식 2는 타당하다.□

보조정리 5-1) EMFG의 초기 마크 벡터의 수는 EMFG의 전체 조건 마킹의 수와 동일하다.

증명) EMFG의 초기 마크 벡터는 정리 5의 식 2와 같으므로 정리는 자명하다.□

예를 들어 그림 25의 EMFG에서 초기 마크 벡터 M_0는 (1, 1, 0, 1, 0, 1, 0, 0, 0, ..., 0), (1, 1, 0, 1, 0, 0, 1, 0, 0, ..., 0) ... 등 8개가 있다.

4.3 수행완료 가능성 알고리듬

UML AD가 수행 완료 가능하려면 정의 4에 의하여 고려 가능한 모든 조건에서 수행 완료 가능하여야 한다. 그리고 UML AD가 어떤 조건에서 수행 완료 가능하다는 것은 InitialNode가 활동 상태에서 시작하여 상태들이 계속 변화하고 최종적으로 ActivityFinalNode가 활동 상태에 있게 되는 경우를 말한다. 여기서 ActivityFinalNode가 활동 상태에 있을 때, 즉 UML AD의 수행이 완료되었을 때는 내부 어떠한 노드도 활동 상태에 있어서는 안된다. 그렇지 않으면 UML AD가 비정상적으로 수행이 완료된 것이다.

UML AD의 어떤 조건에서 수행 완료 가능성을 검사하려면 이를 표현한 EMFG의 동작이 제대로 수행되는지를 검사하면 된다. 따라서 EMFG의 수행 완료 가능성을 다음과 정의한다.

정의 5) EMFG를 초기 마크 벡터 M_0에서 시작하여 동작시켰을 때, 최종적으로 종료 박스의 마크가 존재하는 상태에서 EMFG의 동작이 멈추는 경우, 'EMFG는 M_0에서 수행 완료 가능하다'라고 하고, 가능한 모든 초기 마크 벡터에 대하여 수행 완료 가능하면 'EMFG는 수행 완료 가능하다'라고 한다. 여기서 종료 박스에 마크가 존재할 때는 조건 박스들을 제외한 어떠한 박스에도 마크가 존재해서는 안된다.□

예를 들어 그림 25의 EMFG에서 초기 마크 벡터는 8가지가 있으므로 EMFG의 수행 완료 가능성을 검사하려면 각 초기 마크 벡터에 대하여 EMFG가 수행 완료 가능한지 검사하여야 한다.

그리고 EMFG가 UML AD를 표현한 것이라면 EMFG의 수행 완료 가능성을 사용하여 UML AD의 수행 완료 가능성을 다음과 같이 구할 수 있다.

정리 6) UML AD를 표현한 EMFG가 초기 마크 벡터 M_0에서 수행 완료 가능하면 M_0에 대응되는 조건에서 UML AD도 수행 완료 가능하며, 이 EMFG가 수행 완료 가능하면 이에 대응되는 UML AD도 수행 완료 가능하다.

증명) EMFG는 UML AD를 그대로 표현한 것이므로 EMFG의 동작은 UML AD의 동작과 동일해야 한다. 따라서 정리는 타당하다.□

만약 그림 25의 EMFG가 수행 완료 가능하면 대응되는 그림 24의 UML AD도 수행 완료 가능하며, 그렇지 못하면 UML AD도 수행 완료 가능하지 못할 것이다.

따라서 UML AD의 수행 완료 가능성을 검사하기 위하여 이를 표현한 EMFG의 수행 완료 가능성을 검사하면 되므로 그림 28과 같이 UML AD의 수행 완료 가능성 검사 알고리듬을 구할 수 있다.

UML AD의 수행 완료 가능성 검사 알고리듬

단계 1 : UML AD를 EMFG로 표현한다.

단계 2 : EMFG에서 초기 마크 벡터 M_0를 구한다. 여기서 M_0는 정리 5에 의하여 구할 수 있으며, M_0의 수는 보조정리 5-1에 의하여 구하고 그 결과를 m 개라 하자.

단계 3 : m 개의 M_0에서 각각 EMFG를 동작시켜 수행 완료 가능한지 판단한다. 이때 모든 경우의 M_0에서 EMFG가 수행 완료 가능하면 EMFG가 수행 완료 가능하고 대응되는 UML AD도 수행 완료 가능하다. 그렇지 못하면 EMFG도 수행 완료 가능하지 못하며 대응되는 UML AD도 수행 완료 가능하지 못하다.

그림 28. UML AD의 수행 완료 가능성 검사 알고리듬

그림 28의 알고리듬 적용 예로 그림 24 및 그림 26의 UML AD의 수행 완료 가능성을 다음 절에서 설명한다.

4.4 시뮬레이션

그림 24의 시나리오 1에 대한 UML AD 및 그림 26의 시나리오 2에 대한 UML AD에 대하여 수행 완료 가능성을 시뮬레이션하기로 한다. 우선 그림 24의 UML AD부터 살펴본다.

UML AD의 수행 완료 가능성 검사 알고리듬(이하 알고리듬)의 단계 1을 수행하면 그림 25의 EMFG가 된다. 그림 25에서 마크 벡터 M을 (s, D1, D2, D3, D4, D5, D6, R, SF, EV, TO, AF, PC, CP, AR, Np1, Np2, Np3, Np4, Np5, Np6, Np7, Np8, Np9, e)라 할 때, 알고리듬의 단계 2를 수행하면 표1과 같은 초기 마크 벡터 M_0을 구할 수 있다.

표 1. 그림 25의 초기 마크 벡터

기호	초기 마크 벡터
M_{01}	(1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
M_{02}	(1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
M_{03}	(1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
M_{04}	(1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
M_{05}	(1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
M_{06}	(1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
M_{07}	(1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
M_{08}	(1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)

알고리듬의 단계 3에서 우선 표 1의 M_{01}의 초기 마크 벡터를 적용하여 EMFG를 동작시켜 보기로 한다. 여기서 EMFG의 동작은 EMFG 동작 알고리듬[11]에 따라 수행되는 시뮬레이션 프로그램[18]을 사용하였으며, 시뮬레이션 결과는 그림 29와 같다.

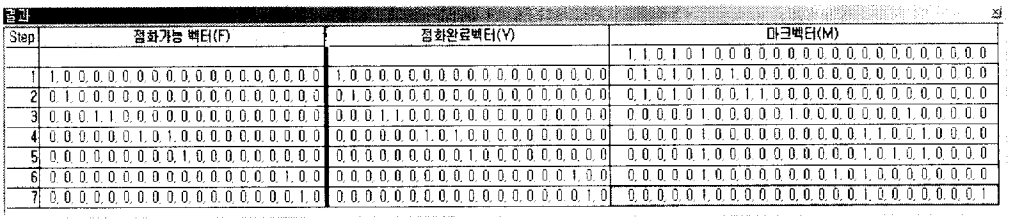


그림 29. 그림 25에 M_{01}을 적용한 시뮬레이션 결과

그림 29의 step 7에서 시뮬레이션이 종료되었으며, 마지막 마크 벡터 M은 (0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1)이 되고, 조건 박스와 종료 박스가 아닌 Np2 박스에 마크가 남아 있음을 알 수 있다. 이로써 M_{01}의 초기 마크 벡터에서 EMFG가 수행 완료 가능하지 못함을 알 수 있다.

표 2는 표 1에 있는 모든 초기 마크 벡터에 대하여 EMFG를 수행시켜 수행 완료 가능성을 조사한 결과이다.

표 2. 그림 25의 시뮬레이션 결과

M	시뮬레이션 결과의 마크벡터	수행완료가능성
M_{01}	(0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1)	X
M_{02}	(0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1)	X
M_{03}	(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1)	O
M_{04}	(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1)	O
M_{05}	(0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1)	O
M_{06}	(0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1)	O
M_{07}	(0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0)	X
M_{08}	(0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0)	X

표 2에서 보는 바와 같이 M_{01}, M_{02}, M_{07}, M_{08} 에서 수행 완료 가능하지 못하므로 그림 25의 EMFG는 수행 완료 가능하지 못하고, 따라서 그림 24의 UML AD도 수행 완료 가능하지 못하다. 즉 고객 불만 처리 워크플로우를 모델링한 그림 24의 UML AD는 잘못 모델링된 UML AD임을 알 수 있다.

다음은 그림 26의 UML AD의 수행 완료 가능성을 시뮬레이션한다.

알고리듬의 단계 1을 수행하면 그림 27의 EMFG가 된다. 그림 27에서 마크 벡터 M을 (s, D1, D2, RO, FO, AA, I, MP, AP, SO, Np1, Np2, CO, e)라 할 때, 알고리듬의 단계 2를 수행하여 초기 마크 벡터 M_0를 구하면 표 3과 같다.

표 3. 그림 27의 초기 마크 벡터

기호	초기 마크 벡터
M_{01}	(1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
M_{02}	(1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)

알고리듬의 단계 3에서 표 3의 M_{01}의 초기 마크 벡터를 적용하여 EMFG를 동작시킨 시뮬레이션 결과는 그림 30과 같다.

결과

Step	점화가능 벡터(F)	점화완료벡터(Y)	마크벡터(M)
			1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
1	1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0	1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0	0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0
2	0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0	0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0
3	0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0	0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0
4	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1

그림 30. 그림 27에 M_{01}을 적용한 시뮬레이션 결과

그림 30의 step 4에서 시뮬레이션이 종료하였으며, 마지막 마크 벡터 M은 (0,0,0,0,0,0,0,0,0,0,0,0,0,1)이 되어 종료 박스에만 마크가 존재하므로 그림 27의 EMFG는 M_{01}에서 수행 완료 가능하다.

표 4는 표 3에 있는 모든 초기 마크 벡터에 대하여 EMFG를 수행시킨 결과이다.

표 4. 그림 27의 시뮬레이션 결과

M	시뮬레이션 결과의 마크벡터	수행완료가능성
M_{01}	(0,0,0,0,0,0,0,0,0,0,0,0,0,1)	O
M_{02}	(0,0,0,0,0,0,0,0,0,0,0,0,0,1)	O

표 4에서 보는 바와 같이 모든 초기 마크 벡터에서 모두 수행 완료 가능하므로 그림 27의 EMFG는 수행 완료 가능하다. 따라서 그림 26의 UML AD도 수행 완료 가능하므로 주문 처리 워크플로우를 제대로 모델링한 UML AD라 할 수 있다.

V. 결 론

최근 기업에서는 경쟁력 제고와 프로세스 혁신을 위한 워크플로우 관리 시스템을 도입하여 고객만족과 생산성 향상을 위하여 노력하고 있다. 이러한 현상으로 현재 기업에서 비즈니스 프로세스 모델링 도구로 UML을 많이 사용하고 있다. UML의 많은 다이어그램 중에서 활동 다이어그램은 업무의 흐름이나 작업단계를 쉽게 알아볼 수 있도록 표현한 다이어그램이다. 이 UML AD은 풍부한 표현력과 사용자 편의성 및 정보 시스템과의 통합성 등의 특성으로 인해 대상 워크플로우를 쉽게 모델링을 할 수 있는 장점이 있으나 분석 능력이 부족하기 때문에 워크플로우의 오류나 수행가능성을 판단하기가 어렵다.

이에 본 논문에서는 UML AD를 EMFG로 표현하여 워크플로우를 모델링한 UML 활동 다이어그램의 수행가능성을 판단하는 방법을 제시하였다. 우선 UML 활동 다이어그램의 구성요소들을 EMFG로 표현하였고 UML 활동다이어그램의 구조에 따라 EMFG로 표현하는 방법을 제안하였다.

또한 본 논문에서는 EMFG의 마크벡터를 이용하여 대상 워크플로우를 모델링한 UML 활동 다이어그램의 수행완료 가능성을 검증함으로써 수행완료 도달 가능 여부를 바로 알 수 있다. 이 수행 완료 가능성의 분석을 통하여 UML 활동 다이어그램이 대상 워크플로우를 제대로 모델링 했는지를 알 수 있다.

향후 UML AD를 변환한 EMFG의 점화가능 벡터와 점화완료 벡터 등을 이용하여 워크플로우의 주된 목적인 워크플로우의 사전 오류 가능성과 교착상태를 판단할 수 있을 것으로 기대된다.

참 고 문 헌

[1] D. Hollingsworth, "Workflow Management Coalition - The Workflow Reference Model", TC00-1003 issue 1.1, 1994.

[2] Work Group1, "Interface 1 : Process Definition Interchange Process Model", Workflow Management Coalition Specification, TC 016, 1998.

[3] Rik Eshuis, Roel Wieringa, "Verification Support for Workflow Design with UML Activity Graphs", ICSE'02, pp.166-176, 2002.

[4] OMG, "OMG Unified Modeling Language Specification(version 2.0), part 10-Activites", http:/www.omg.com, 2004.

[5] Hammer, M. and Champy, J.(1993), Reengineering the Corporation: a Manifesto for Business Revolution, Harper Business, New York.

[6] Mayer, R. J., Menzel, C. P., Painter, M. K., deWitte, P. S., Blinn, T. and Perakath, B.(1995), Information Integration for Concurrent Engineering IDEF3 Process Description Capture Method Report, KBSI System Inc., Texas.

[7] Sheer, A.-W., ARIS Business Process Modeling, Springer-Verlag, Berlin. 1999.

[8] Van der Aalst, W. M. P., Woflan: A Petri Net Based Workflow Analyzer, Systems Analysis, Modeling, Simulation, 35(3), pp.245-357, 1999.

[9] 한관희, "UML 활동 다이어그램의 페트리넷 변환을 통한 워크플로 분석", IE Interfaces Vol. 17, No.2, pp.200-207, 2004.

[10] 이동익, "페트리 넷 이론의 기초", 정보처리학회지, Vol.2, No.2, 1995.

[11] 김희정, 여정모, 서경룡, "EMFG의 개선된 동작해석 알고리즘", 정보처리학회논문지A, 제9-A권 제3호, pp.371-378, 2002.

[12] 한관희, 황태일, "UML/XML 기반의 비즈니스 프로세스 정의 도구", IE Interfaces Vol. 16, No.2, pp.156-166, 2003.

[13] 한관희, 황태일, "표준 워크플로우 정의 데이터를 산출하는 UML 기반 프로세스 모델링 도구 개발", 한국경영과학회/대한산업공학회 2003 춘계공동학술대회, 2003.

[14] 여정모, "마크흐름선도의 확장", 부산대학교 대학원 석사학위 논문, 1982.

[15] 허후숙, 여정모, "워크플로우의 EMFG 모델링과 분석", 한국정보처리학회논문지D 제10-D권 제7호, pp.1189-1196, 2003.

[16] 김소연, 이강수, "워크플로우 모형화 및 관리시스템", 정보처리 제3권 제5호, pp.18-30, 1996.

[17] 허후숙, "워크플로우의 EMFG 모델링과 분석", 부경대학교 교육대학원 석사학위 논문, 2003.

[18] 이태훈, "EMFG 시뮬레이터 설계 및 구현", 부경대학교 대학원 석사학위 논문, 2004.

[19] 여정모, "이산 제어시스템 설계를 위한 확장된 마크흐름선도의 동작해석", 정보처리 논문지 Vol.5, No.7, pp.1896-1907, 1998.

[20] 홍현기, "워크플로우 시스템 구축을 위한 프로세스 지향적인 방법론에 관한 연구", 한독경상학회 경상논집 Vol.21, pp.221-242, 2000

[21] 슈물러, 조세핀, "(초보자를 위한)UML 객체지향설계", 인포북, 1999

감사의 글

대학 학부과정을 마치고, 회사생활을 시작하면서 시간이 지나자 무엇인지 모를 허기와 여러 가지 이유들로 또 다시 배움의 길인 교육대학원에 입학하게 되었습니다. 처음 입학할 때는 졸업이란 선배들만의 영광인 것 같았는데 어느덧 제가 졸업을 하게 되었습니다. 2년 반 동안 힘들고 어려운 적도 많았지만 또 다른 많은 경험을 할 수 있는 기회였습니다. 제가 대학원을 무사히 마치게 될 수 있도록 제게 힘이 되고 버팀목이 되어주신 많은 분들께 지면으로나마 감사의 마음을 전하고 싶습니다.

모든 면에 부족한 저를 이끌어주시고 가르쳐주시고, 항상 성실과 열정이 가득한 모습으로 가르침을 주신 여정모 지도교수님께 깊은 감사를 드립니다. 바쁘신 중에도 부족한 저의 논문을 심사해 주신 윤성대 교수님, 박홍복 교수님께 감사의 마음을 드립니다. 그리고 언제나 많은 가르침을 주셨던 박만곤 교수님, 박승섭 교수님, 김창수 교수님, 김영봉 교수님, 이경현 교수님, 정순호 교수님께도 감사드립니다.

학교생활 중에 가장 많은 시간들을 같이 지낸 연구실 식구들에게 감사의 마음을 전합니다. 듬직한 모습으로 조언과 충고로 많은 도움을 주신 홍석선배, 열정과 강인한 체력으로 늘 열심히 사시는 모습 보여주시는 명희 샘~, 입학 동기로서 많은 의지가 되어 주신 허선자 선생님, 그리고 제 논문에 많은 관심 보여주신 박교환 선생님, 예쁘고 착한 우리 연구실 학부생들(은정, 재욱, 윤영)....
많은 시간 학교생활을 같이 하지는 못했지만 많은 도움을 준 연구실 선배들, 늘 조언과 위로를 함께 해주고 도움을 준 친구같은 후숙언니, 성실과 책임의 모습을 보여주는 정숙언니, 연구실 선배이자 착하고 이쁜 동생인 안니, 옥과 홍 ,현주, 입학 동기로서 많은 도움과 의지가 되고 있는 동기들~
대학원 다니면서 일과 학교로 바빠서 많은 연락 못하고 지내도 항상 응원해주고 지지해 주는 영원한 친구 찬숙, 성정숙, 미정, 혜림, ...

특히나 멀리 떨어져있는 가족, 언니, 오빠들께는 교육 대학원을 다니는 동안 항상 미안한 마음과 감사하는 마음을 동시에 갖고 지냈습니다. 바람에 잎새가 흔들려도, 이슬비 한방울이 내려도 늘 제 걱정이신 부모님! 언제나 항상 한결같이 늘 믿어주시고 정신적 지주가 되어주시는 사랑하는 부모님께 진심으로 깊은 감사의 마음을 전합니다.