EMFG를 이용한 UML 활동 다이어그램의 수행가능성 평가

여 정 모[†] · 이 미 순[††]

요 약

UML(Unified Modeling Language)은 객체지향 모델링을 위한 언어이다. UML에 포함되어 있는 AD(Activity Diagram; 활동 다이어그램)는 워크플로우 시스템의 모델링에 적합하지만 수행 과정의 평가가 수학적이지 못하므로 수행 과정을 직관적으로 파악해야 하는 단점이 있다. 그러나 EMFG(Extended Mark Flow Graph)는 이산적인 시스템의 모델링에 적합할 뿐 아니라 수행 과정을 수학적으로 평가할 수 있다. 따라서 본 연구에서는 EMFG를 사용하여 모델링된 AD의 수행 동작을 평가하려고 한다. 우선 AD를 EMFG로 변환하는 방법에 대하여 제안하고, 변환된 EMFG의 수행을 평가함으로써 모델링된 AD의 수행완료 가능성을 평가한다. 그리고 컴퓨터 시뮬레이션을 통하여 이를 입증한다. 제안한 알고리듬은 대규모 워크플로우 시스템을 모델링한 AD의 수행완료 가능성도 쉽게 평가할 수 있는 장점이 있다.

키워드 : EMFG, UML, AD, 수행가능성

The Performance-ability Evaluation of an UML Activity Diagram with the EMFG

Jeong Mo Yeo[†] · Mi Soon Lee[††]

ABSTRACT

Hardware and software codesign framework called PeaCE(Ptolemy extension as a Codesign Environment) was developed. It allows to express both data flow and control flow which is described as fFSM which extends traditional finite state machine. While the fFSM model provides lots of syntactic constructs for describing control flow, it has a lack of their formality and then difficulties in verifying the specification. In order to define the formal semantics of the fFSM, in this paper, firstly the hierarchical structure in the model is flattened and then the step semantics is defined. As a result, some important bugs such as race condition, ambiguous transition, and circulartransition can be formally detected in the model.

Key Words : EMFG(Extended Mark Flow Graph), UML(Unified Modeling Language), AD(Activity Diagram), Performance-ability

1. 서 론

최근의 급속한 경영 환경 변화 아래서 생존하기 위해 각 기업들은 프로세스 혁신이나 BPR(Business Process Reengineering) 등을 통해 경쟁력을 확보하려 총력을 기울이고 있으며[13], 이를 성공적으로 수행하기 위해 많은 기업에서 워크플로우 관리 시스템의 도입이 확산되고 있는 추세이다[9].

워크플로우 시스템은 업무의 동적인 흐름을 정확히 표현해야 하는데, 이를 모델링하기 위하여 IDEF3(Integrated DEFinition3), ARIS(ARchitecture of integrated Information Systems) EPC(Event driven Process Chain), UML(Unified Modeling Language) AD(Activity Diagram), 페트리넷 및 EMFG(Extended Mark Flow Graph) 등을 많이 사용한다[6-8]. 특히 이 중에서 UML AD(이하 AD라 한다)는 객체지향 모델링 표준으로서 널리 사용되고 있으며, 기업 활동의 동적 흐름 표현, 이벤트 표현, 객체지향 표준 준수 및 사용자 편이성이라는 측면에서 기업프로세스를 모델링하는데 적합하다. 그러나 AD를 사용하여 워크플로우 시스템을 모델링하였을 때, 수행되는 과정을 수학적으로 평가하는 방법이 없어 직관적으로 파악해야 하기 때문에 규모가 큰 경우에는 분석이 어려워지는 단점이 있다. 이를 해결하기 위한 방법으로 수학적 형식론에 기반하여 시스템의 행위를 수학적으로 기술할 수 있는 페트리 넷을 이용한 연구[9]가 진행되고 있지만 아직 미비한 실정이다.

그러나 EMFG는 이산 시스템의 모델링에 적합할 뿐 아니라 수행되는 동작을 수학적으로 평가할 수 있어 컴퓨터 시뮬레이션도 가능하다. 따라서 본 연구에서는 워크플로우를 모델

† 정 회 원 : 부경대학교 전자컴퓨터정보통신공학부 교수
†† 준 회 원 : 부경대학교 교육대학원 전산교육전공
논문접수 : 2005년 9월 1일, 심사완료 : 2005년 11월 25일

링한 AD를 분석하기 위해, 먼저 AD를 EMFG로 변환하고, 이를 사용하여 AD의 수행완료 가능성을 평가하고자 한다.

EMFG는 박스(box), 트랜지션(transition), 아크(arc)들로 구성되는 마크를 갖는 방향성 선도로서 동기나 비동기의 이산제어 시스템을 설계하여 구현하거나 분석하는데 적합[14, 19]할 뿐 아니라 EMFG 동작 알고리듬[11]에 따라 시뮬레이션하면 수학적으로 쉽게 분석할 수 있으므로 대규모 시스템의 모델링에도 손쉽게 사용할 수 있는 장점이 있다.

2. UML AD의 EMFG 표현

2.1 UML과 AD

UML은 Rumbaugh의 OMT(Object Modeling Technology)의 방법론, Booch의 OOAD(Object Oriented Analysis and Design) 방법론 및 Jacobson의 OOSE (Object Oriented Software Engineering) 방법론 등의 장점을 통합하여 만든 모델링 언어로서, 객체지향 분석 및 설계 방법론의 표준 지정을 목표로 제안되었다[21].

AD는 UML의 여러 다이어그램 중의 하나인 흐름도형태로서, 활동으로부터 활동으로 전달되는 제어 흐름과 객체간 제어흐름 등을 표현하고, 처리와 결정 및 분기 등을 표현할 수 있으므로 업무 처리과정이나 오퍼레이션에서 처리되는 일들을 효과적으로 표현할 때 유용하게 사용된다.

AD는 그래픽 노드(Node)들과 그래픽 경로(Flow)들 및 그래픽 요소(Element)들로 표현[4]된다. 그래픽 노드에는 Action Node, ActivityFinalNode, InitialNode, DecisionNode, ForkNode, JoinNode, MergeNode, Send-SignalAction, Accept EventAction 등이 있으며, 이들은 워크플로우에서 실제 단위 업무를 나타내는 동적활동이나 그 동적활동을 지원하는 정적활동을 의미한다. 그래픽 경로에는 ControlFlow, ObjectFlow 등이 있으며, 이들은 각 활동 간의 흐름이나 객체간의 흐름을 나타낸다. 그래픽 요소는 빈번하게 사용되지 않거나 예외적인 상황에서 사용되므로 워크플로우를 대상으로 하는 본 연구에서는 고려하지 않는다.

2.2 용어 정의

본 연구에서 취급하는 AD의 노드들과 경로들을 다음과 같이 정의한다.

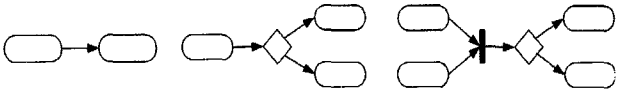
정의 1. AD의 노드들 중에서 상태를 의미하는 Action Node, ObjectNode, Activity FinalNode 및 InitialNode를 상태노드(state node)라 하고, 제어의 의미를 나타내는 Decision Node, MergeNode, ForkNode 및 JoinNode를 제어노드(control node)라 한다. □

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

AD 구조는 정의 1의 노드들과 정의 2의 경로들로 구성되고, 이를 상태 흐름에 따라 구분하여 다음과 같이 정의한다.

정의 3. AD에서 상태노드와 상태노드가 경로로 결합된 구조를 SS(State-State) 구조, 상태노드와 상태노드 사이에 제어노드가 경로로 결합된 구조를 SCS(State-Control-State) 구조, 상태노드와 상태노드 사이에 2개 이상의 제어노드들이 경로로 결합된 구조를 SMCS(State-Multi-Control-State) 구조라 한다. □

AD의 각 구조에 대한 예를 (그림 1)에 나타내었다.



(그림 1) AD에 나타나는 구조의 예

SS 구조는 어떤 상태에서 다른 상태로 변화됨을 의미하고, SCS 구조와 SMCS 구조는 어떤 상태에서 제어되어 다른 상태로 변화됨을 의미한다. 그리고 각 구조에서 앞쪽의 상태노드는 ActionNode, ObjectNode 및 InitialNode 등이 될 수 있으며, 뒤쪽의 상태노드는 ActionNode, ObjectNode 및 ActivityFinalNode 등이 될 수 있다.

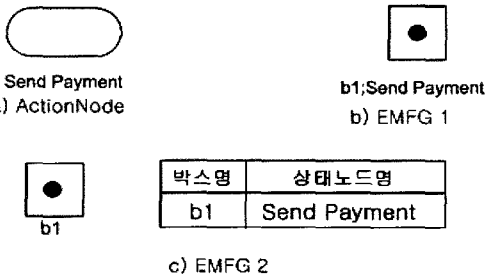
2.3 AD의 EMFG 표현

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

우선 상태노드의 상태를 EMFG의 박스 상태와 대응시킨다면, 다음과 같이 상태노드를 EMFG로 표현할 수 있다.

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

예를 들어 (그림 2) (a)는 ActionNode인 'Send Payment' 노드를 (그림 2) (b)또는 (그림 2) (c)와 같이 EMFG의 박스



(그림 2) 상태노드의 EMFG 표현 예

b1로 표현할 수 있으며, 'Send Payment' 노드가 활동 중일 때는 b1에 마크가 존재하게 된다.

특히 InitialNode와 ActivityFinalNode는 AD의 시작과 끝에 나타나므로 이에 대응되는 EMFG 박스를 시작박스(s : start box)와 종료박스(e : end box)라 한다.

다음은 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. AD에서 SS 구조의 EMFG 표현 : 이 구조에서 입력 노드를 I, 출력 노드를 O라 할 때, 이 구조는 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 3) (a)의 ActionNode와 ObjectNode로 구성되는 SS 구조를 EMFG로 표현하면 (그림 3) (b)와 같다. (그림 3) (a)에서 SS 구조의 상태 흐름은 (그림 3) (b)에서 EMFG의 마크 이동으로 파악된다.

a) SS 구조 b) a)의 EMFG

(그림 3) SS 구조의 EMFG 표현

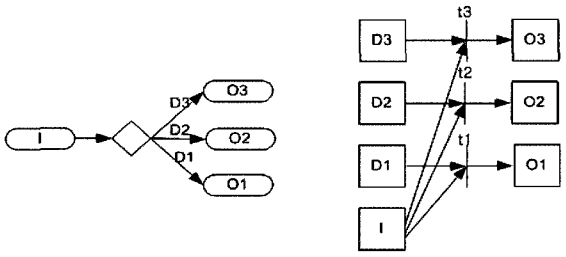
SCS 구조는 포함된 제어노드의 제어에 의해 상태 변화가 일어나므로 제어노드의 제어 방법에 따라 다르게 상태를 변화시킨다. 따라서 제어노드의 종류에 따라 EMFG로 표현하는 방법도 서로 다르게 된다.

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

규칙 3. AD에서 DecisionNode를 가진 SCS 구조의 EMFG 표현 : 이 구조에서 DecisionNode가 n개의 조건을 가진 경우, 입력 노드를 I, DecisionNode의 각 조건을 Di, Di에 연결된 출력 노드를 Oi라 할 때, 이 구조는 다음 순서에 따라 EMFG로 표현한다. 여기서 i는 1 이상이고 n 이하인 정수이다.

1) I 및 각 Oi를 규칙1에 따라 각각 해당 박스로 표현한다.
2) Di를 각각 하나의 박스로 표현한다. 여기서 각 박스를 조건박스라 하며, 조건 Di의 만족여부는 대응되는 조건박스의 마크 존재유무를 의미한다.
3) I에 대응되는 박스와 Di에 대응되는 박스를 입력으로 가지고, Oi에 대응되는 박스를 출력으로 가지는 트랜지션 ti를 각각 구성한다. 여기서 각 박스들은 ti에 일반아크로 연결된다. □

(그림 4) (a)의 DecisionNode를 가진 SCS 구조를 EMFG로 표현하면 (그림 4) (b)와 같다. (그림 4)에서 보는 바와 같이 AD의 조건에 따라 상태 변화가 일어나는 모습을 EMFG에서도 쉽게 파악할 수 있다.



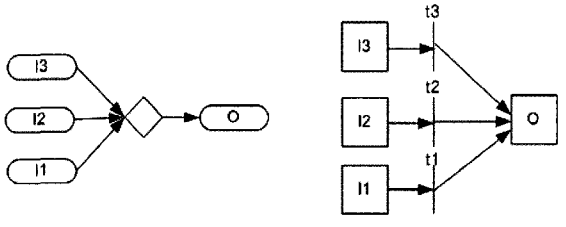
a) DecisionNode를 가진 SCS 구조 b) a)의 EMFG

(그림 3) DecisionNode를 가진 SCS 구조의 EMFG 표현

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

규칙 4. AD에서 MergeNode를 가진 SCS 구조의 EMFG 표현 : 이 구조에서 MergeNode가 n개의 입력 노드를 가지는 경우, 각 입력 노드를 Ii, 출력 노드를 O라 할 때, 이 구조는 다음 순서에 따라 EMFG로 표현한다. 여기서 i는 1 이상이고 n 이하인 정수이다.

1) Ii 및 O를 규칙1에 따라 각각 해당 박스로 변환한다.
2) Ii에 대응되는 박스를 입력으로, O에 대응되는 박스를 출력으로 가지는 트랜지션 ti를 구성한다. 여기서 각 박스들은 트랜지션 ti에 일반아크로 연결된다.□



a) MergeNode를 가진 SCS 구조 b) a)의 EMFG

(그림 5) MergeNode를 가진 SCS 구조의 EMFG 표현

(그림 5) (a)의 MergeNode를 가진 SCS 구조를 EMFG로 표현하면 (그림 5) (b)와 같다. (그림 5)에서 보는 바와 같이 AD에서 여러 상태 중에서 하나 이상이 만족되면 다음 하나

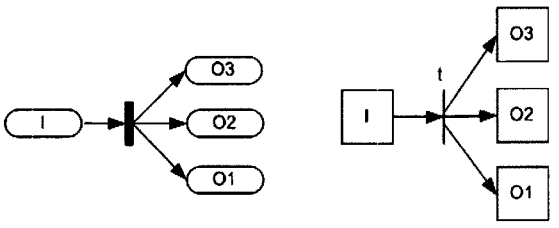
의 상태로 변하게 되는 모습을 EMFG에서도 그대로 쉽게 파악할 수 있다.

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

규칙 5. AD에서 ForkNode를 가진 SCS 구조의 EMFG 표현 : 이 구조에서 ForkNode가 n개의 출력 노드를 가지는 경우, 입력 노드를 I, 각 출력 노드를 Oi라 할 때, 이 구조는 다음 순서에 따라 EMFG로 표현한다. 여기서 i는 1 이상이고 n 이하인 정수이다.

1) I 및 Oi를 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. □

(그림 6) (a)의 ForkNode를 가진 SCS 구조를 EMFG로 표현하면 (그림 6) (b)와 같다. (그림 6)에서 보는 바와 같이 AD에서 하나의 상태에서 두개 이상의 상태로 변하게 되는 모습이 EMFG에서도 파악될 수 있다.



a) ForkNode를 가진 SCS 구조 b) a)의 EMFG

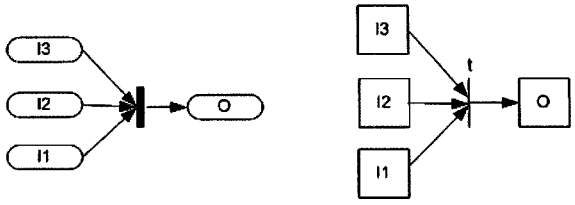
(그림 6) ForkNode를 가진 SCS 구조의 EMFG 표현

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

규칙 6. AD에서 JoinNode를 가진 SCS 구조의 EMFG 표현 : 이 구조에서 JoinNode가 n개의 입력 노드를 가지는 경우, 각 입력 노드를 Ii, 출력 노드를 O라 할 때, 이 구조는 다음 순서에 따라 EMFG로 표현한다. 여기서 i는 1 이상이고 n 이하인 정수이다.

1) Ii 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 일반아크로 연결된다. □

(그림 7) (a)의 JoinNode를 가진 SCS 구조를 EMFG로 표현하면 (그림 7) (b)와 같다. (그림 7)에서 보는 바와 같이 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 7) JoinNode를 가진 SCS 구조의 EMFG 표현

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

정리 1. AD에서 상태노드 N1, 두 개의 제어노드 C1과 C2, 상태노드 N2가 직렬로 구성되는 SMCS 구조가 있다. 이 구조에서 C1과 C2 사이에 ActionNode인 가상노드(pseudo node) NP를 두어 C1과 C2를 경로로 연결하였을 때, N1과 C1 및 NP로 구성되는 SCS 구조를 SCS1, NP와 C2 및 N2로 구성되는 SCS 구조를 SCS2라 하자. 이때 SMCS 구조의 동작은 SCS1과 SCS2로 구성된 구조의 동작과 서로 동일하다.

증명) SMCS 구조에서 C1의 상태는 C2의 입력으로 직접 사용되어 C2의 상태를 변화시킨다. 그리고 SCS1의 C1의 상태는 그대로 NP에 전달되고, 이 NP의 상태는 SCS2의 NP의 상태와 동일하므로 NP의 상태가 그대로 C2에 입력되어 C2의 상태를 변화시킨다. 따라서 이 동작은 SMCS 구조의 동작과 동일하므로 정리는 타당하다. □

보조정리 1-1. AD에서 상태노드 N1, n개의 제어노드가 있을 때 각 제어노드를 Ci(1 <= i <= n), 상태노드 N2가 직렬로 구성되는 SMCS 구조가 있다. 이 구조에서 C1과 C2 사이에 가상노드 NP1, C2와 C3 사이에 가상노드 NP2, ..., Cn와 N2 사이에 가상노드 NPn를 두어 각 가상노드를 인접하는 제어노드들과 경로로 연결하였을 때, N1과 C1 및 NP1로 구성되는 SCS 구조를 SCS1, NP1과 C2 및 NP2로 구성되는 SCS 구조를 SCS2, ..., Cn과 NPn 및 N2로 구성되는 SCS 구조를 SCSn라 하자. 이때 SMCS 구조의 동작은 SCS1, SCS2, ..., SCSn로 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.

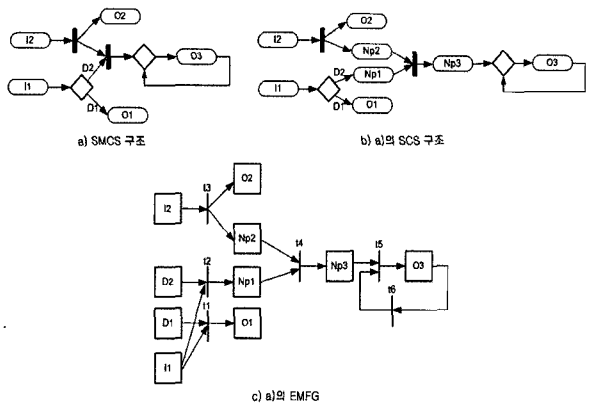
규칙 7. AD에서 SMCS 구조의 EMFG 표현 : 이 구조는 다음 순서에 따라 EMFG로 표현한다.

1) 이 구조의 동작과 동일하도록 SMCS 구조를 가상노드들

이 포함된 여러 개의 SCS 구조로 변환한다. 여기서의 변환은 보조정리 1-1이 적용되도록 변환되어야 한다.

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

(그림 8) (a)는 여러 종류의 제어노드가 연결되어 표현되는 SMCS 구조이며, 규칙7에 따라 (그림 8) (a)를 (그림 8) (b)와 같이 가상노드 Np를 가진 SCS 구조로 변환한 후 (그림 8) (c)와 같이 EMFG로 표현할 수 있다.

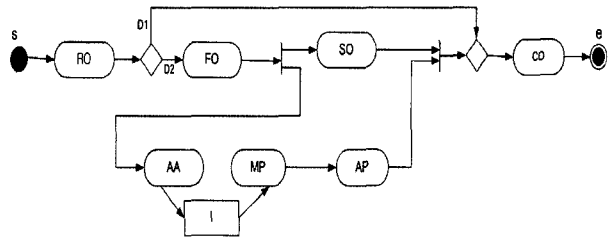


(그림 8) SMCS 구조의 EMFG 표현 예 3

2.4 AD의 EMFG 표현 예

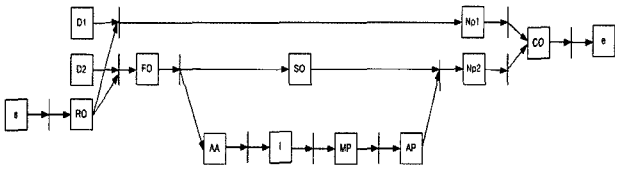
본 절에서는 2.3절의 규칙들에 의해 AD의 예를 EMFG로 표현한다.

예 1은 OMG(2004)의 주문처리 워크플로우[4]로서, 이를 AD로 표현하면 (그림 9)[4]와 같다.



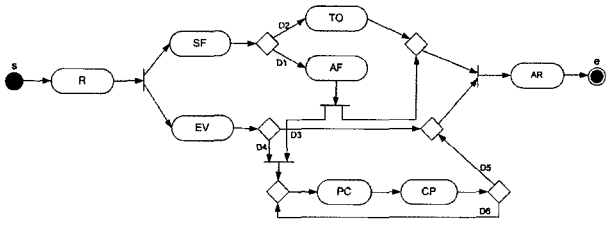
(그림 9) 예1의 AD

(그림 9)의 AD를 EMFG로 표현하면 (그림 10)과 같다.



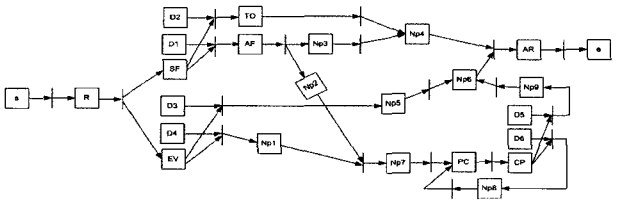
(그림 10) (그림 9)의 EMFG 표현

예 2는 고객 불만 처리 워크플로우[9]로서, 이를 AD로 표현하면 (그림 11)[9]과 같다.



(그림 11) 예2의 AD

규칙 1에서 규칙 7까지의 규칙을 사용하여 (그림 11)의 AD를 EMFG로 표현하면 (그림 12)와 같다. 여기서 AD의 노드들과 조건들의 명칭은 EMFG 표현에서 그대로 사용하였으며, Np1에서 Np9는 AD의 SMCS 구조에서 SCS 구조로 변환되면서 생성된 가상 노드이다.



(그림 12) (그림 11)의 EMFG 표현

두 예에서 보는 바와 같이 AD는 쉽게 EMFG로 표현되며, 그 동작도 대응되는 AD의 동작과 동일하다. 이는 직관적으로도 파악되지만, 정의와 변환 규칙 및 정리 등에서 명백하다.

3. AD를 표현한 EMFG의 분석

3.1 AD의 수행완료 가능성

워크플로우의 동작을 AD로서 모델링하는 경우, 워크플로우의 동작을 쉽게 파악할 수 있다. 그러나 AD의 동작이 올바르게 수행되는지 평가할 수 있는 수학적인 방법이 없지만, EMFG의 동작은 수학적으로 평가 가능하다. 따라서 본 연구에서는 AD를 변환한 EMFG를 사용하여 원래 AD의 동작을 평가하려고 한다.

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

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

만약 어떤 조건 D에서 AD가 수행완료가능하지 않으면, 조건 D에서 수행을 시작하여 종료하지 못할 것이므로 이에 대응되는 워크플로우의 동작도 제대로 완료되지 않을 것이

다. 워크플로우를 AD로 모델링할 경우는 논리적으로 합당하게 모델링해야 할 뿐만 아니라 가능한 모든 조건에서 수행 완료 가능하도록 모델링해야 한다.

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

3.2 AD를 표현한 EMFG의 초기 마크 벡터

EMFG를 동작시키기 위해서는 초기 마킹을 부여해야 한다. AD를 표현한 EMFG의 초기 마킹은 다음 조건을 만족한다.

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

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

정리 3. EMFG의 초기 마킹을 결정할 때, AD에서 하나의 DecisionNode를 표현한 EMFG의 조건 박스들은 오직 하나의 조건 박스에만 마크가 존재하여야 한다.

증명) 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. □

EMFG의 조건 박스(들)에 마크를 부여하는 것을 EMFG의 조건 마킹이라 한다.

보조정리 3-1. AD에서 n개의 조건을 가진 하나의 Decision Node를 표현한 EMFG의 조건 마킹의 수는 n개이다.

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

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

증명) AD가 수행되려면 모든 Decision-Node마다 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. m개의 DecisionNode를 가지는 AD에서 i번째 DecisionNode를 Di, Di를 구성하는 조건의 수를 Ni라 할 때, 이와 같은 AD를 표현한 EMFG의 전체 조건 마킹 수 S는 (식 1)과 같다.

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

증명) Di를 표현한 EMFG의 조건 박스들의 조건 마킹 수는 보조정리 3-1에 따라 Ni개이고, 이는 각 Decision Node를 표현한 EMFG의 조건 박스들에도 동일하게 적용되므로 EMFG에서 전체 조건 마킹 수 S는 (식 1)과 같다. 따라서 정리는 타당하다. □

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 2)와 같다.

$$M_0 = (1, \text{조건마킹상태}, 0, 0, \ldots, 0) \qquad \text{(식 2)}$$

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

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

증명) EMFG의 초기 마크 벡터는 (식 2)와 같으므로 초기 마크 벡터의 수는 조건마킹 상태에 따라 결정되고, 조건 마킹의 수는 (식 1)과 같으므로 정리는 자명하다. □

예를 들어 (그림 10)의 EMFG에서 초기 마크 벡터 M_0는 (1,1,0,0,0,0,0,0,0,0,0,0,0,0)과 (1,0,1,0,0,0,0,0,0,0,0,0,0,0) 2개가 있다.

3.3 수행완료 가능성 알고리듬

AD가 수행 완료 가능하려면 정의 4에 의하여 가능한 모든 조건에서 수행 완료 가능하여야 하고, 각 조건에서 수행 완료 가능하려면 InitialNode가 활동 상태에서 시작하여 상태들이 계속 변화하고 최종적으로 ActivityFinalNode가 활동 상태에 있어야 한다. 여기서 ActivityFinalNode가 활동 상태에 있을 때, 즉 AD의 수행이 완료되었을 때는 내부 어떠한 노드도 활동 상태에 있어서는 안 된다. 그렇지 않으면 AD가 비정상적으로 수행이 완료된 것이다.

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, 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 안 된다. □

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

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

AD의 수행 완료 가능성을 검사하기 위하여 이를 표현한 EMFG의 수행 완료 가능성을 검사하면 되므로 (그림 13)과 같이 AD의 수행 완료 가능성 검사 알고리듬을 구할 수 있다.

단계 1 : AD를 EMFG로 표현한다.
단계 2 : EMFG에서 가능한 모든 초기 마크 벡터를 구한다. 구한 초기 마크 벡터를 M_{01}, M_{02}, ..., M_{0m}이라 하자.
단계 3 : EMFG에 M_{0i}를 적용하여 동작시킨다. 동작 결과가 수행완료 가능하면 다음 단계로 가고, 그렇지 못하면 단계 5로 간다. 여기서 i의 초기값은 1이다.
단계 4 : i를 1 증가시키고 i <= m 이면 단계 3으로 간다.
단계 5 : i가 m+1이면 EMFG가 수행 완료 가능하고, 대응되는 AD도 수행 완료 가능하다. 그렇지 못하면 EMFG도 수행 완료 가능하지 못하며, 대응되는 AD도 수행 완료 가능하지 못하다.

(그림 13) AD의 수행완료가능성 검사 알고리듬

3.4 시뮬레이션

(그림 9)의 예 1에 대한 AD의 수행 완료 가능성을 시뮬레이션하기로 한다. AD의 수행 완료 가능성 검사 알고리듬(이하 알고리듬)의 단계 1을 수행하면 (그림 10)의 EMFG가 된다. (그림 10)의 마크 벡터 M을 (s, D1, D2, RO, FO, AA, I, MP, AP, SO, Np1, Np2, CO, e)라 할 때, 알고리듬의 단계 2를 수행하면 <표 1>과 같은 초기 마크 벡터 M_{01}과 M_{02}를 구할 수 있다.

〈표 1〉 (그림 10)의 초기 마크 벡터

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

알고리듬의 단계 3에서 우선 <표 1>의 M_{01}의 초기 마크 벡터를 적용하여 EMFG를 동작시켜 보기로 한다. 여기서 EMFG 동작 알고리듬[11]에 따라 수행되는 시뮬레이션[18]을 사용하였으며, 시뮬레이션 결과는 (그림 14)와 같다.

(그림 14) (그림 10)의 M_{01} 시뮬레이션 결과

(그림 14)의 step 4에서 시뮬레이션이 종료하였으며, 마지막 마크 벡터 M은 (0,0,0,0,0,0,0,0,0,0,0,0,1)이 되어 종료 박스에만 마크가 존재하므로 (그림 10)의 EMFG는 M_{01}에서 수행 완료 가능하다.

<표 2>는 <표 1>에 있는 모든 초기 마크 벡터에 대하여 EMFG를 수행시킨 결과이다.

〈표 2〉 (그림 10)의 시뮬레이션 결과

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

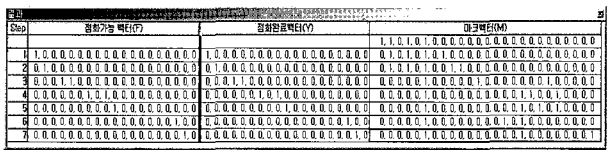
<표 2>에서 보는 바와 같이 모든 초기 마크 벡터에서 모두 수행 완료 가능하므로 (그림 10)의 EMFG는 수행 완료 가능하다. 따라서 (그림 9)의 AD는 수행 완료 가능하게 되어 주문 처리 워크플로우를 올바르게 모델링하였다고 할 수 있다.

다음은 예 2에 대한 AD에 대하여 수행 완료 가능성을 시뮬레이션한다. 알고리듬의 단계 1을 수행하면 (그림 12)의 EMFG가 된다. (그림 12)에서 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 <표 4>와 같은 초기 마크 벡터들을 구할 수 있다.

〈표 4〉 (그림 11)의 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)
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)
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)
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)
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)
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)
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)
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)

알고리듬의 단계 3에서 우선 <표 4>의 M_{01}의 초기 마크 벡터를 적용하여 시뮬레이션한 결과는 (그림 15)와 같다.



(그림 15) (그림 12)에 M_{01}을 적용한 시뮬레이션 결과

(그림 15)의 step 7에서 시뮬레이션이 종료되었으며, 마지막 마크 벡터 M에서 조건 박스와 종료 박스가 아닌 Np2 박스에 마크가 남아 있게 되어 M_{01}의 초기 마크 벡터에서 EMFG가 수행 완료 가능하지 않다.

<표 5>는 <표 4>에 있는 모든 초기 마크 벡터에 대하여

〈표 5〉 (그림 12)의 시뮬레이션 결과

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

EMFG를 수행시켜 수행 완료 가능성을 조사한 결과이다.

<표 5>에서 보는 바와 같이 M_{01}, M_{02}, M_{07}, M_{08}에서 수행 완료 가능하지 못하므로 (그림 12)의 EMFG는 수행 완료 가능하지 않아 (그림 11)의 AD도 수행 완료 가능하지 못하다. 즉 고객 불만 처리 워크플로우를 모델링한 (그림 11)의 AD는 잘못 모델링된 결과라 할 수 있다.

4. 결 론

본 연구에서는 다음과 같은 결론을 얻었다.

1) AD(UML의 활동다이어그램)를 EMFG로 변환하는 방법을 제안하였다. 변환된 EMFG의 동작은 원래의 AD의 동작과 동일하다.
2) AD의 수행완료가능성을 평가하기 위한 알고리듬을 제안하였다. 이 알고리듬은 AD가 변환된 EMFG의 동작을 시뮬레이션함으로써 수행되는데, 이를 컴퓨터 시뮬레이션함으로써 입증하였다.
3) 모델링된 AD는 수행을 평가하는 수학적인 방법이 없는데 반하여, 제안한 알고리듬은 컴퓨터 시뮬레이션이 가능하다. 이로 인하여 대규모 시스템을 모델링한 AD의 수행가능성을 쉽게 파악할 수 있게 되었다.

향후 연구방향은 AD를 변환한 EMFG의 점화가능 벡터와 점화완료 벡터 등을 분석하여 연구하면 워크플로우의 오류 위치 및 교착상태도 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 석사학위논문, 1982.
[15] 허후숙, 여정모, "워크플로우의 EMFG 모델링과 분석", 정보처리학회논문지D, 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, "EMFG시뮬레이터 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)UML객체지향설계", 인포북, 1999.

여 정 모

e-mail : yeo@pknu.ac.kr
1980년 동아대학교 전자공학과 졸업
1982년 부산대학교 대학원 전자공학과 (공학석사)
1993년 울산대학교 대학원 전자 및 전산기공학과(공학박사)
1986년~현재 부경대학교 전자컴퓨터정보통신공학부 교수
관심분야 : 전자상거래, 데이터베이스

이 미 순

e-mail : angdollee@naver.com
2000년 부경대학교 전자계산학과(학사)
2005년 부경대학교 교육대학원 전산교육전공(교육학석사)
관심분야 : 전자상거래, 데이터베이스