액티비티 다이어그램을 이용한 워크플로우 시스템 모델링

(Workflow System Modeling using Activity Diagram)

고 은 정 † 이 상 영 †† 유 철 중 ††† 장 옥 배 †††
(Eun-Jung Ko) (Sang-Young Lee) (Cheol-Jung Yoo) (Ok-Bae Jang)

요 약 UML(Unified Modeling Language) 액티비티 다이어그램은 시스템의 동적인 측면을 표현하는데 적합하기 때문에 비즈니스 프로세스와 워크플로우를 모델링하는데 유용하게 사용된다. 그러나 워크플로우 시스템에서 중요시 되는 시멘틱 정보 표현의 경우 OMG가 제공하는 UML 액티비티 다이어그램에 대한 시멘틱 정보를 표현하기 위한 지침만으로는 정확한 시멘틱 정보의 표현이 어렵다. 이에 본 논문에서는 워크플로우 시스템의 특성에 맞도록 ASM(Abstract State Machine) 시멘틱을 확장한 후 액티비티 다이어그램에 적용하여 워크플로우 시스템을 모델링하는 방안을 제시한다. 이와 같은 ASM 시멘틱을 기반으로 하는 정형적인 시멘틱에 대한 정확한 정의를 통하여 보다 효율적인 워크플로우 모델링이 가능하다.

키워드 : 워크플로우, 모델링, 시멘틱, 액티비티 다이어그램

Abstract UML activity diagram is useful to model business process and workflow by reason of its suitability to present dynamic aspect of system. However it is difficult to present precise semantics which is taken as important in workflow system with the guide provided by OMG to the UML activity diagram.

This paper suggests workflow system modelling methodology by applying ASM semantics to the activity diagram after extending its semantics to correspond to workflow system characteristics. Through the exact definition to formal sematics based on ASM it is possible to effectively model workflow.

Key words : workflow, modeling, semantics, activity diagram

1. 서 론

UML(Unified Modeling Language)은 소프트웨어 모델링에 대한 OMG(Object Management Group)의 표준으로, 여기에는 소프트웨어 시스템의 구조와 행위를 분석하고 표현하기 위한 여러 종류의 다이어그램들이 있다[1]. 이중에 액티비티 다이어그램(activity diagram)은 시스템의 동적인 측면을 묘사하기 위한 다이어그램이다[2].

기존에 시스템의 동적인 측면을 묘사하는 모델링 방법 중 액티비티 다이어그램과 페트리네트(Petri-Net)를 이용한 모델링 방법이 멀티 쓰레드(multi-threaded)되거나 비동기적 시스템을 표현하기에 효율적인 방법이다[2]. 따라서 워크플로우(workflow) 모델링에 액티비티 다이어그램과 페트리네트가 널리 사용되고 있다. 그러나 페트리네트의 경우 워크플로우와 같은 복잡한 시스템에 적용시 실제 비즈니스 프로세스를 모델링하는데 복잡해지고 난해해지는 단점이 있다[3,4]. 반면 액티비티 다이어그램은 페트리네트보다 이벤트 지향적인 행위나 데이타를 표현하기에 적합하다[5,6]. 액티비티 다이어그램의 주된 목적은 비즈니스 프로세스(business process)와 소프트웨어 프로세스(software process)를 모델링하는 것이다[7,8]. 즉, 시스템의 동적인 측면을 표현하는데 사용되는 액티비티 다이어그램은 비즈니스 프로세스와 워크플로우를 모델링하는데 적합하다[6,9]. 즉 액티비티 다이어그램은 여러 객체에 속하는 액션(action)들의 흐름을 묘사하며, 도메인에서의 객체 사이의 종속성을 묘사함으로써 워크플로우 모델링에 많이 사용된다[10].

워크플로우 모델링은 반드시 전문가가 아니더라도 이

· 본 논문은 한국과학재단의 지역대학우수과학자 지원연구에 의한 것임 (과제번호: R05-2003-000-12236-0)
† 비 회 원 : 전북대학교 컴퓨터과학과
juko@chonbuk.ac.kr
†† 정 회 원 : 남서울대학교 보건행정학과 교수
sylee@nsu.ac.kr
††† 종신회원 : 전북대학교 컴퓨터과학과 교수
cjyoo@chonbuk.ac.kr
okjang@chonbuk.ac.kr
논문접수 : 2003년 9월 15일
심사완료 : 2005년 4월 7일

해할 수 있고 작업하기 쉬운 보편성을 가질 수 있도록 개발되어야 한다. 또한 모델링 기법 자체에는 정형적인 의미를 내포하고 있어야 하고 분석 가능하여야 한다[11]. 워크플로우를 이러한 관점에서 모델링한다면 워크플로우 수행 중 발생가능한 예외사항 및 다양한 변화 등에 대처가능하다.

비록 OMG가 액티비티 다이어그램에 대한 문서의 시멘틱(semantics) 정보를 제공하고 있지만 정확한 시멘틱의 부족으로 인하여 모호해지는 단점이 있다[12]. 지금까지 액티비티 다이어그램에서의 정형화된 시멘틱 제공은 미비하며 비공식적인 OMG 시멘틱으로는 워크플로우 적용 시 적합하지 않다[13]. 즉, 현재의 UML은 정형적이고 공통적으로 동의된 시멘틱에 대한 명세가 부족하다[14]. 따라서 이러한 시멘틱 명세의 부족은 산출물을 모델링하는 것에 대한 정확한 이해를 하는데 시간을 소비하기 때문에 실제적으로 UML의 사용을 방해하게 된다. 즉 워크플로우 모델링을 효율적으로 수행하기 위해서는 수학적인 의미를 내포하도록 정형적인 시멘틱으로 표현되어 분석가능하여야 한다.

기존의 UML 액티비티 다이어그램을 정형화하기 위한 여러가지 시멘틱 방법들이 있다. 예를 들어 액티비티 다이어그램의 시멘틱을 표현하기 위해 Pi-calculus[12]나 FSP(Finite State Processes)[10] 등을 채택하는 방법들이 있다. 또한 OCL(Object Constraint Language)를 사용하여 모델 요소 사이의 제약사항을 기술하기도 한다. 그러나 이러한 방법들은 액티비티 다이어그램의 동적인 행위 자체를 표현하기에는 부족하다[15].

이에 본 논문에서는 이와 같은 시멘틱 부족에 의한 문제를 해결하기 위해서 액티비티 다이어그램의 시멘틱 부분을 ASM(Abstract State Machine)[16,17] 시멘틱 방법을 적용하여 제시한다. 또한 이를 워크플로우 시스템 모델링 기법에 적용하여 확장시킨다. 즉, UML 액티비티 다이어그램에 대한 ASM 시멘틱 형태를 정의하는 구체적인 접근 방법을 제시한다.

본 논문의 나머지 부분은 다음과 같이 구성된다. 2장은 관련 연구로 워크플로우 모델링과 액티비티 다이어그램에서의 시멘틱을 중심으로 한 연구들에 대하여 제시하고, 각 시멘틱들의 특성과 정의들에 대해 살펴본다. 그리고 3장에서는 ASM 시멘틱 기법을 이용한 액티비티 다이어그램의 시멘틱을 정의하고, 4장에서는 워크플로우 시스템에 ASM 시멘틱을 확장시켜 적용하고 확장된 액티비티 다이어그램을 제시한다. 마지막으로 5장에서는 사례 연구를 제시하고 기존의 연구와 비교 평가하고, 6장에서는 결론 및 향후 연구 방향을 제시한다.

2. 관련 연구

워크플로우 기술은 지난 수십 년 동안 많은 발전이 있었고, 이미 현존하는 워크플로우 시스템 역시 매우 다양하다. 그러나 이러한 워크플로우 시스템을 적용한 사례들 대부분이 성공하지 못한 이유로는 시스템을 표현하는 모델들이 너무 제한적이고 유연성이 부족했기 때문이다[11]. 이러한 문제점들을 극복하기 위해서는 발생가능한 예외상황을 대처할 수 있도록 지원하며 조직 내의 프로세스 및 액티비티들을 효율적으로 표현할 수 있는 방법으로 워크플로우를 모델링하는 것이 필요하다.

2.1 워크플로우 모델링

모델링은 복잡한 시스템을 수학적으로 분석하고 시스템과 사용자들 간의 의사전달 및 시스템 자체의 시뮬레이션 연구를 위해서 이해하고 개발하는데 있어 유용하게 이용되고 있다. 그리고 워크플로우 모델링은 조직이나 작업 그룹에서의 업무 환경과 프로세스를 적절히 표현하는 것을 말한다. 워크플로우에서는 조직의 모습을 업무(task), 참여자(participant), 역할(role), 액티비티(activity), 자료(resource) 및 저장소(repository) 등의 형태로 표현한다.

지금까지 시스템의 동적인 측면을 모델링하는 것에 대한 여러 연구들이 수행되어 왔다. 즉 상태 다이어그램(state diagram), 상태 차트(state chart), 협력 다이어그램(collaboration diagram), 순차 다이어그램(sequencing diagram), 액티비티 다이어그램, 페트리네트 등을 이용한 연구들이 수행되어 왔다. 이중에서 액티비티 다이어그램과 페트리네트가 워크플로우 모델링에 널리 사용되어 왔다[2].

2.2 액티비티 다이어그램에 대한 시멘틱 표현 기법

워크플로우 명세는 어떻게 워크플로우 시스템이 행동하는가를 묘사한다. 그러므로 워크플로우 시스템의 명세와 관련하여 액티비티 다이어그램의 시멘틱을 정의하고 의미를 부여할 필요가 있다. 지금까지 액티비티 다이어그램을 정형화하기 위한 다양한 시멘틱 방법들이 제안되었다. 이와 같은 액티비티 다이어그램의 시멘틱을 표현하기 위한 방법에는 OCL, Pi-calculus, FSP, ASM 등이 있다. 먼저 OCL의 경우 액티비티 다이어그램 모델 요소 사이의 제약사항 등을 정의하는데 도움을 준다. 그러나 동적이고 행위적인 모델을 표현하기에 너무 다변적이고 비즈니스 프로세스 특성을 모델링하는데 필요한 제어흐름을 표현하지 못하는 단점을 가진다[15]. 그리고 Pi-calculus의 경우 동시에 발생하는 시스템을 표현하는 모델링 기법으로 프로세스 사이의 상호작용을 표현하기에 적합하다. 여기서는 LTS(Label Transition System)에 대한 시멘틱을 정의하고 있다. Pi-calculus의 두 가지 기본 개념으로는 이름과 프로세스가 있다. 먼저 이름(name)은 채널(포트), 변수 및 데이타를 말하

며, 프로세스(process)는 시스템에서 엔터티(entity)를 표현한다. 그러나 Pi-calculus 시멘틱의 경우 객체 흐름이 모델링 자체에 포함되어있지 않다는 단점을 가진다[12]. Pi-calculus에서는 환경과 함께 상호작용하는 엔터티들이 프로세스로서 모델링된다. 따라서 액션노드와 서브 액티비티노드를 프로세스로 모델링한다. 이들은 이벤트를 주고 받는 포트(port)를 통하여 상호작용한다. 즉 프로세스는 입력포트를 통해 프로세스를 시작하는 이벤트를 받고 액션을 실행한다. 액션이 완료될 때 프로세스는 액션 완료 이벤트를 보내고 다음 프로세스의 입력포트를 실행시킨다. 그리고 의사(pseudo) 상태를 나타내는 프로세스는 액션을 실행하지 않고 비즈니스 프로세스의 액티비티들 사이의 흐름을 제어하는 기능으로 사용된다. 마지막 전이(transition)는 다른 상태노드에 상응하는 프로세스의 입력포트와 시작노드에 상응하는 프로세스의 출력포트를 연결하는 오퍼레이터(operator)로 모델링된다. 표 1은 이러한 모델 요소에 대한 pi-calculus의 표현 방법을 나타낸다.

표 1 모델 요소들에 대한 Pi-calculus의 표현

Model elements	Pi-calculus
Action state	Process expression
Subactivity states	Process corresponding to another activity diagram
Pseudo-state node	Process expression
Transition(edge)	Linking operator

표 2 액티비티 다이어그램으로부터 FSP로 매핑

UML Activity Diagram	FSP
begin	BEGIN=(start → BEGIN)
End	→STOP
ControlFlow	→
ActionState	→a
ActivityState	P=(a→P)
StereoType	P=(a→S), S=(c→P)
JoinBar	
OR-JOIN	P=(a→JoinBar|b→JoinBar|c→JoinBar), JoinBar=(d→P).
XOR-JOIN	P=(when cond1 a→JoinBar|b→JoinBar|c→JoinBar), JoinBar=(d→P).
AND-JOIN	P1=(a→d→P1). P2=(b→d→P2). P3=(c→d→P3). ||P=(P1||P2||P3).
SplitBar	
OR	P=(d→SplitBar), SplitBar=(a→P|b→P|c→P).
AND	D=(d→D). P1=(d→a)→A). P2=(d→b). P3=(d→c). ||P=(D||P1||P2||P3).
Branch	
XOR	P=(d→Branch), Branch=(when cond1=true a →P| when cond2=true b→P | when cond3=true c→P
Proxy	Synchronised actions

또한 액티비티 다이어그램의 시멘틱을 표현하기 위한 방법 중 하나인 FSP의 경우에는 UML 액티비티 다이어그램을 정형화함으로써 시멘틱 방법을 제공하지만 적용 방법 자체가 난해한 단점이 있다[10]. 이러한 FSP는 LTS로서 시스템을 모델링하는데 사용되는 정형화 기법이다. 여기에는 프로세스 행위를 모델링하기 위한 구성 요소들을 있다. 표 2는 이러한 구성 요소를 통한 UML 액티비티 다이어그램으로부터 FSP로의 매핑을 나타낸다.

마지막으로 본 논문에서 적용하는 시멘틱 방법인 ASM은 Gurevich에 의하여 십여년 전에 제안된 이래 많은 소프트웨어 시스템을 명세하고 검증하는데 효율적으로 사용되어 왔다[18]. ASM은 어느 알고리즘이든지 적절한 추상 레벨에서 모델링이 가능하다는 장점을 가진다[19]. 또한 명확성, 정확성, 이해가능성, 실행가능성 및 보편성 측면에서 장점이 있다[16,17].

이에 본 연구에서는 워크플로우 모델링에 ASM을 사용하여 액티비티 다이어그램에 대한 시멘틱을 정의한다. 즉 동적인 시스템의 행위를 묘사하는 액티비티 다이어그램에 대한 적절한 시멘틱을 제공하는 것을 목표로 한다.

3. ASM 시멘틱 기법을 이용한 액티비티 다이어그램의 시멘틱 정의

3.1 액티비티 다이어그램

액티비티 다이어그램에서는 순차적인 제어 흐름뿐만 아니라 병렬적으로 수행되는 활동과 분기가 이루어지는 대안들에 대해서도 표현가능하다. 액티비티 다이어그램은 기본적으로 액티비티, 전이, 의사결정 지점(decision point)과 동기화 막대(synchronization bar) 등으로 구성된다[9].

모든 액티비티 다이어그램에서는 실행되는 프로세스들을 정의한다. 즉 프로세스의 결정, 반복, 병렬 및 랜덤한 행위 등을 표현한다. 액티비티 다이어그램은 주로 시스템의 동적인 측면을 모델링하는 것으로 시스템의 동적인 측면을 모델링할 때 협력하는 액터들을 고려함으로써 이들의 액티비티에 중점을 두고 모델링한다. 다음 표 3은 액티비티 다이어그램 구성 요소의 기호(signature) 및 정의를 나타낸다[9].

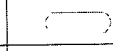
3.2 ASM 시멘틱 정의

본 논문에서는 액티비티 다이어그램의 구성요소를 ASM 시멘틱 기법을 이용하여 재정의한다. 다음은 UML 액티비티 다이어그램의 각각의 구성 요소 중 액션노드와 분기노드의 기호에 ASM 시멘틱 규칙을 적용한 것이다.

먼저 액션노드는 다음과 같이 표현한다.

node(in, A, out, isDynamic, dynArgs, dnyMult)

표 3 액티비티 다이어그램 구성요소의 기호(signature)와 정의

구성요소	기 호	정 의
initial node	●	다끈트 값으로 시작을 나타냄
final node	◉	제어 흐름이 종료된 것을 가리킴
action node		액티비티의 액션을 표현하는 것으로 실행(작업)은 중단할 수 없음
Decision node		대안적인 경로를 나타내고 하나의 입력 전이와 두개 이상의 출력 전이를 가지며, 각각의 출력 전이는 가드로 알려는 부울(boolean) 표현과 연관이 있음
fork/join bar		하나의 흐름을 두개로 분할하거나 동시에 합성하는 제어의 흐름을 각각 두개나 그 이상의 동시에 발생하는 제어의 흐름으로 동기화함
transition		하나의 액션이나 액티비티로부터 다른 액션이나 액티비티까지 가는 정로를 표시
swimlane		액티비티에 대하여 책임있는 조직적인 단위를 표현하는 각각의 그룹은 자신만의 고유한 이름을 가짐

여기서 파라미터 in과 out은 입력과 출력 전이를 나타내며, A는 원자의 액션노드를 나타낸다. 이러한 액션노드는 이외에도 isDynamic, dynArgs 및 dynMult 파라미터를 갖는다. 먼저 A가 dynMult에 해당하는 시간에 실행된다면 원자의 액션 A는 isDynamic하다고 할 수 있다. 또한 각각의 시간은 L_i의 독립변수를 가지며, 이는 객체 $\{L_1, \ldots, L_n\}$의 연속인 dynArgs의 집합으로부터 나온 것이다. 다음 그림 1은 이러한 액티비티 다이어그램의 구성요소인 액션 노드를 나타낸다.

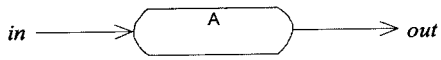


그림 1 액션노드의 기호

액션노드를 ASM 규칙으로 표현하면 다음과 같다. 일반적으로 작동중인 에이전트는 다음의 액션을 결정하기 위해서 활동중인 전이의 목표(target)를 파악하고 연관된 가드가 참인지 체크하여야 하며 방해가 발생하지 않도록 하여야 한다. 이러한 작동중인 에이전트를 여기서는 currTarget로 표현한다. 이러한 currTarget이 액션노드이고 dynArgs의 수가 동적인 dynMult와 대응된다면, A는 일반적으로 각각의 독립변수 목록으로 실행된다. 그러나 만일 액션노드가 동적이 아니라면 A는 그대로 수행된다. 다음은 액션노드에 대한 ASM 시멘틱을 나타낸다.

if *currTarget is node(in, A, out, isDynamic, dynArgs, dynMult)*
then if $\neg$ *isDynamic* then *A*
elseif $[dynArgs] = dynMult$ then
for all $L_i \in dynArgs$
$A(L_i)$
active : = $\rightarrow$ *out*

다음으로 분기를 나타내는 노드는 다음과 같이 나타낸다.

node(in, cond, out)

여기서 파라미터 in과 out은 입력과 출력 전이를 나타내며, cond는 부울(boolean) 표현을 나타낸다. 만일 분기노드에 제어가 위치하는 경우, 연관된 가드 조건이 만족되고 방해되는 이벤트가 발생하지 않는다면 조건은 평가되고 제어는 참 조건에 연관된 전이를 통과하게 된다. 분기노드를 ASM의 규칙으로 표현하면 다음과 같다.

if *currTarget is* $node(in, (cond_i)_{i \le n}, (out_i)_{i \le n})$
then
if $cond_1$ then *active* : = $\rightarrow out_1$
.....
elseif $cond_n$ then *active* : = $\rightarrow out_n$

이와 같은 ASM 시멘틱에 해당하는 분기노드의 기호는 그림 2와 같다.

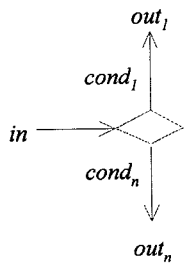


그림 2 분기노드의 기호

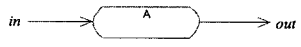
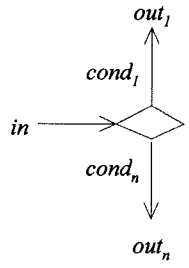
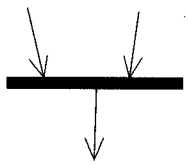
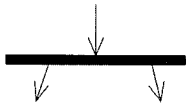
아울러 이와 같은 방법으로 액티비티 각각의 구성요소에 ASM 시멘틱을 적용하여 나타내면 표 4에서와 같이 나타낼 수 있다. 여기서는 액티비티 각각의 구성요소에 해당하는 기호와 시멘틱을 보여준다. 시작노드와 종료노드는 파라미터가 없는 특별한 액션노드로 간주하여 표에서는 생략하였다.

4. 워크플로우 시스템에 대한 ASM 시멘틱의 적용

4.1 워크플로우 특성을 고려한 확장된 시멘틱

워크플로우란 한 조직체 내에서 발생하는 여러단계의 복잡하고 다양한 비즈니스 업무 흐름을 정의하고 이들의 수행을 위한 효율적인 상호 작업 환경을 제공하는 자동화된 서비스를 의미한다[20]. 즉, 워크플로우는 절차상의 규칙집합을 중심으로 문서, 정보 또는 작업으로 구성된 하나의 액션이 한 참여자로부터 다른 참여자까지 거치는 동안의 전체 또는 부분에 해당하는 비즈니스 프

표 4 액티비티 구성요소에 해당하는 ASM 기호와 시멘틱

노 드	기 호	시 멘 틱
action node		if *currTarget is node(in, A, out, isDynamic, dynArgs, dynMult)* then if ¬ *isDynamic* then *A* elseif [*dynArgs*] = *dynMult* then for all $L_i \in$ *dynArgs* $A(L_i)$ *active* : = → *out*
decision node		if *currTarget is node(in,* $(cond_i)_{i\le n}$, $(out_i)_{i\le n}$) then if $cond_1$ then *active* : = →out_1 ..... elseif $cond_n$ then *active* : = →out_n
join bar		if *active* = *out*(*node*(*in*, $D_1, \ldots, D_m$, *out*))) & *mode*(*Self*) = *waiting* & $\forall a_i \in$ *SubAgent*(*Self*): *mode*(a_i) = *running* & *active*(a_i) = *finalArc*(D_i) then do forall $a_i \in$ *SubAgent*(*Self*) *delete*(a_i) *mode*(*Self*): = *running*
fork bar		if *currTarget is node*(*in*, $D_1, \ldots, D_m$, *out*) then extend *AGENT* with $a_1 \ldots a_n$ do forall 1≤i≤m *activate*(a_i, D_i) *parent*(a_i): = *Self* *mode*(*Self*) : = *waiting* *active*: = *out*

로세스에 대한 자동화를 의미한다. 워크플로우 관리 시스템의 대부분은 이러한 비즈니스 프로세스를 중심으로 이루어진다. 즉 워크플로우 시스템의 목적은 사람들로 구성된 조직에 의해서 수행되는 비즈니스 프로세스 고유의 협업적인(collaborative) 특성에 있다. 이러한 워크플로우를 관리하기 위해서는 비즈니스 프로세스를 만족하기 위한 자동화된 조정, 통제 및 업무간의 통신 등이 필요하다. 여기서 협업(collaboration)은 워크플로우에서의 중요한 특성으로 일반적으로 세 가지 관점으로 구분된다[21]. 즉 협업에서 수반되는 모임 사이의 정보를 교환하기위한 기본적인 능력인 통신(communications), 이러한 모임에 의해서 수행된 작업을 지시하고 계획하는 조정(coordination), 협력적으로 비즈니스 프로세스를 수행하는 협력(cooperation) 등이 있다.

워크플로우의 구성요소는 비즈니스 프로세스를 나타내는 일련의 물리적 또는 논리적 단위인 액티비티들과 이를 수행하기 위한 참여자(participant), 그리고 액티비티들 간에 전달되는 문서 또는 정보들이 있다[22]. 이러한 워크플로우 시스템의 형태만으로는 시기적절성을 수반하는 시스템의 카테고리를 모델링하기에 적절치 않으며, 비즈니스 프로세스를 모델링하는데에도 적합하지 않다. 특히 실시간 비동기적 시스템을 모델링하려면 다음 사항들을 고려하여야 한다. 즉 환경에서 예기된 변경과 예외, 동적인 변경사항, 전체 워크플로우 모델을 개발하기 위하여 제약된 워크플로우의 양 및 시간적 조절(timing) 요인 등이 모델링되어야 한다.

전통적인 워크플로우 모델링에 포함되는 내용은 다음과 같다. 즉 시스템 또는 서브시스템 조직을 묘사하는 조직적인 단위, 역할, 프로세스 내에서 작업의 조작을 묘사하는 액티비티, 액티비티의 수행동안에 발생하는 사건을 묘사하는 이벤트, 작업을 수행하기 위해 요구될 기계나 도구, 장치를 묘사하는 자원, 그리고 시스템의 목적을 달성하기 위해 몇 가지 액티비티를 포함하는 프로세스 등을 포함한다[23]. 또한 액티비티들이 임의의 방법으로 실행될 수 없기 때문에 워크플로우 모델은 조건 또는 제약사항 등을 포함한다. 즉 사전 조건과 사후 조건의 두 가지 형태로 워크플로우 모델에서 적용될 수 있다. 먼저 사전 조건은 워크플로우를 실행하기 위해서 만족되어져야만 하며, 사후 조건은 워크플로우를 끝내기 위해서 만족되어져야만 한다.

또한 워크플로우 모델링에서는 적격의 사용자 또는 애플리케이션 기능에 대한 액티비티의 할당을 통하여 액티비티를 누가 제어하는가, 액티비티의 입력, 출력 및 작업 수행에 대하여 요구된 데이타와 제어 정보, 액티비티의 사전 및 사후 조건, 어떤 액티비티들이 액티비티를 완료하기 위해서 요구되는가 등에 대한 정보가 필요하다[24].

이와 같은 워크플로우의 특성을 바탕으로 워크플로우 모델링에서의 필요한 정보는 다음과 같다.

• 데이타(data)

데이타는 액션에서 생산되거나 사용될 데이타를 표현하는 것으로, 입력 데이타는 활동에 대해 사용될 데이타이며, 출력 데이타는 활동에 의해서 생산된 데이타를 말한다.

• 조건(conditions)

조건은 액션의 조건을 표현하는 것으로 사전 조건은 만족될 필요가 있는 조건으로 활동이 규정될 수 있으며, 사후 조건은 활동의 종료 시 만족되어져야만 하는 조건이다.

• 자원(resource)

활동을 수행하기위해 요구되는 것들을 표현하는 것으로, 작업을 수행하기위해 요구될 기계, 도구 및 장치를 묘사한다.

• 참여자(participant)

작업을 수행하는 자원에 해당하는 참여자는 누가 활동을 수행하는가를 표현하며 조직의 일부분으로 액티비티와 관련된 역할을 수행한다. 여기에는 액터(actor), 에이전트(agent), 플레이어(plyer), 사용자(user), 역할 수행자(role player) 등이 있다.

• 액션(action)

액션은 액터나 액터들에 의하여 수행되며, 그것은 수동 액션(manual action) 또는 자동 액션(automatic action)일 수 있다. 수동 액션은 액터에 의하여 수동적으로 행해지는 행동이며, 자동 액션은 워크플로우 관리 시스템에 의하여 자동적으로 행해지는 행동을 말한다. 액션에서 생산되거나 사용될 입력 데이타와 출력 데이타, 수동액션인지, 자동 액션인지를 나타내는 액션의 상태를 나타내는 정보들을 액션노드에서 확장 한다. 또한 액션 상태에 의존하는 사전 조건과 사후 조건을 확장한다.

• 타이밍(timing)

타이밍은 시기적절성을 뜻하는 것으로 워크플로우의 액션이 수행되는 동안의 시간이나 데드라인(deadline) 등의 시간적인 요인이 필요하다.

그리고 기존 액션노드의 ASM 시멘틱을 확장하여 다음과 같이 표현한다.

node(in, in_data, A, out, out_data, isDynamic, dynArgs, dnyMult, action_type, time)

in_data, out_data의 파라미터는 앞에서 설명된 액션에서 사용되는 입력 데이타와 액션에서 생산되는 출력 데이타를 나타낸다. 출력 데이타는 액션노드 A가 수행된 후에 생성되어야 한다. 마지막으로 action_type의 파라미터는 액션의 수행 형태가 수동으로 수행되는지 자동으로 수행되는지를 나타내는 것으로, 만일 이것이 수동으로 수행되어진다면 이는 참여자를 반드시 수반하여야 한다. 마지막으로 time 파라미터는 만일 시간이 제한 시간을 초과한다면 종료하도록 한다.

다음은 확장된 액션노드에 대한 AMS 시멘틱을 나타낸 규칙이다.

```
if currTarget is node(in, in_data, A, out, out_data,
 isDynamic, dynArgs, dynMult, action_type, time)
then if ¬ isDynamic then A
    elseif [dynArgs] = dynMult then
            for all L_i ∈ dynArgs
    A(L_i)
    elseif time ≥ deadline
            then active = exit()
    elseif action_type is manual then
            active = participaint
    active : = → out
```

그리고 확장된 사전 조건노드는 다음과 같이 표현한다.

node(PRE, out, isExistent)

사전 조건이 있는 경우에 사전 조건을 수행하며, 사전 조건이 참이라면 out 전이를 활성화 시키게 된다. 액션을 수행하기 전에 사전 조건을 만족하는 경우만 액션노드를 실행할 수 있다.

```
if currTarget is node(PRE, out, isExistent)
then if isExistent then PRE
     elseif PRE = true then
          active : = → out
```

또한 확장된 사후 조건노드는 다음과 같이 표현한다.

```
node(in, POST, isExistent)
```

사후 조건은 활동의 종료 시 만족되어져야만 하는 조건이다. 액션을 실행한 후에 사후 조건이 있는 경우에 사후 조건을 수행하며, 만일 사후 조건이 참이라면 다음 액션노드를 실행할 수 있도록 한다.

그리고 자원노드는 자원 활동을 수행하기 위해 요구되는 것들을 표현하는 것으로, 작업을 수행하기 위해 요구되는 기계, 도구 및 장치를 묘사한다. 자원노드는 시작노드와 종료노드와 마찬가지로 파라미터가 없는 특별한 액션노드로 간주하도록 한다.

```
if currTarget is node(in, POST, isExistent)
then if isExistent then POST
     elseif POST = true then
          active : = nextnode(in, in_data, A, out, out_data,
isDynamic, DynArgs, dynMult, action_type, time)
```

4.2 확장된 액티비티 다이어그램의 표기법

여기서는 확장된 ASM 시멘틱의 내용을 액티비티 다이어그램에 적용시켜 액티비티 다이어그램을 확장시킨다. 확장된 내용들을 액티비티 다이어그램에서는 다음과 같이 나타낸다.

먼저 액션노드의 확장된 부분들을 나타내는 액티비티 다이어그램의 표기법은 그림 3과 같다.

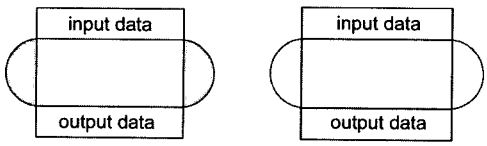


그림 3 액션 표기

그림에서 보는 바와 같이 액션 타입이 수동인지 자동인지를 구별하기 위하여 수동인 경우에는 색이 채워진 액션노드로, 자동인 경우에는 색이 채워지지 않은 노드로 표현한다. 액션노드에 필요한 입력 데이타와 액션노드로부터 나오는 출력 데이타는 위아래의 사각형 부분에 각각 입력 데이타와 출력 데이타를 나타낸다. 또한 시간에 대한 항목을 액션 속성으로 지정할 수 있도록 하며, 가시적으로 표기법을 확장하지는 않는다. 즉 액션노드의 타입과 시간은 속성으로 모두 지정할 수 있도록 한다.

액션을 표기하는데 도움을 주는 사전 조건과 사후 조건의 표기법은 그림 4에서와 같이 오각형 모양으로 나타낸다. 여기서 사전 조건과 사후 조건은 모드 액션노드와 연관 관계를 맺으며 점선의 화살표로 액션노드와의 의존 관계를 표현한다. 왼쪽에 있는 노드가 사전 조건을 나타내며 오른쪽노드는 사후 조건을 나타낸다.

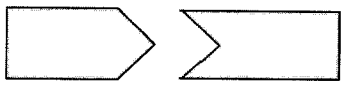


그림 4 사전 조건과 사후 조건 표기

또한 자원은 특별한 액션노드로 그림 5와 같이 표기한다.

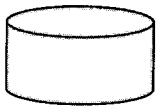


그림 5 자원 표기

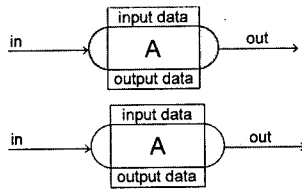
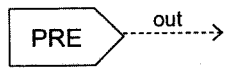
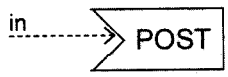
그리고 나머지 표기법은 UML 액티비티 다이어그램에서와 같다. 이렇게 확장된 표기법은 워크플로우를 모델링하는데 좀더 유연성을 가지도록 도와주며, 워크플로우 특성을 반영할 수 있는 장점이 있다. 다음 표 5는 제시하는 표기법으로 표현된 액티비티의 구성요소에 해당하는 확장된 ASM 시멘틱을 나타낸다.

5. 사례 연구 및 평가

대표적인 조직 내부의 워크플로우에는 상품주문 처리절차(order processing), 상품구매 처리절차(purchasing), 휴가 신청 처리절차(holiday requests), 출퇴근 및 근무시간 처리절차(time sheets), 지출결의서 처리절차(expense reports), 생산주문서 처리절차(production procedures), 그리고 기타 고객지원 관련 처리절차(customer support) 등이 있다. 또한 최근에 관심이 증가하고 있는 전자거래, 공급망 관리, 전자물류, 전자조달, 전자정보, 웹 서비스 등은 프로세스를 기반으로 하는 조직간 워크플로우 또는 프로세스의 대표적인 예이다[25].

본 논문에서는 상품을 주문하고 구매하는 소매업에서의 워크플로우에 적용한다. 소매업에서는 인간 시스템(통신판매, 마케팅, 구매, 선적 업무요원 등)과 함께 자동화된 시스템(마케팅과 창고 시스템을 교류하는 POS (Point of Sale) 시스템 등)이 존재한다. 다양한 자동화된 시스템과 인간 시스템이 협업하는 비즈니스 프로세스를 액티비티 다이어그램을 이용하여 모델링한다. 특히

표 5 확장된 액티비티 구성요소에 해당하는 기호와 시멘틱

노 드	기 호	시 멘 틱
action node		if *currTarget is node(in, in_data, A, out, out_data, isDynamic, dynArgs, dynMult, action_type, time)* then if ¬ *isDynamic* then *A* elseif [*dynArgs*] = *dynMult* then for all $L_i \in dynArgs$ $A(L_i)$ elseif *time* ≥ *deadline* then *active* = *exit*() elseif *action_type is manual* then *active* = *participaint* *active* : = → *out*
pre-condition node		if *currTarget is node(PRE, out, isExistent)* then if *isExistent* then *PRE* elseif *PRE* = *true* then *active* : = → *out*
post-condition node		if *currTarget is node(in, POST, isExistent)* then if *isExistent* then *POST* elseif *POST* = *true* then *active* : = *nextnode(in, in_data, A, out, out_data, isDynamic, DynArgs, dynMult, action_type, time)*

본 논문에서는 선적회사가 고객이 우편 주문했던 품목을 반품하는 워크플로우 사례를 적용한다.

워크플로우의 작업은 선적회사가 회사에게 품목을 반품하는 것을 요구하면서 시작된다. 품목 반품을 요구하기 이전에 품목은 고객이 반품하기를 요구한 것이며, 이미 팔린 것이어야만 한다. 이와 같은 조건을 만족한다면 선적회사는 품목 반품 요청 작업을 수행하게 된다. 그 다음 회사에서 반품 번호를 얻은 후, 선적회사에서 품목을 선적하여 반품하게 된다. 여기에서 반품 번호를 얻는 액션에서 반품 번호를 입력 데이타로 넣어야 하며, 품목을 선적하는 경우 입력 데이타는 반품될 품목의 번호이고, 조건으로 품목이 선적될 준비가 미리 되어있어야 한다. 또한 품목이 선적이 된 후에는 품목이 반품되어야 한다. 반품하게 된 품목은 회사에서 받게 된다. 이때 입력 데이타로 품목의 반품 번호를 받게 되며 출력 데이타로 저장소를 출력한다. 저장소에서 품목을 받게 되므로 저장소 노드와 관련이 있으며, 정확하게 품목이 받아졌는지 확인한 후에 재고를 다시 채우게 된다. 만일 정확한 품목이 받아지지 않았다면 다시 선적회사에서 물건을 선적하도록 한다. 정확한 품목이 받아지게 되면 다시 재고에 품목이 채워진다. 재고가 채워지면 입력 데이타로 품목의 반품 번호를 입력받고, 출력 데이타로 다시 부여받은 품목의 번호를 출력하게 된다. 이러한 활동이 종료될 때 품목은 사용가능하도록 상태가 변경된다. 마지막으로 경리과에서 예금 계좌로 입금하고 동시에 고객에게 메일을 발송함으로써 반품처리가 종료된다. 예금 계좌로 입금하기 위하여 만족되어야할 조건은 품목이 재고에 존재해야만 하며, 품목의 가격을 입력받아서 돈을 넣어주고 출력 데이타로 돈을 넣어준 영수증에 대한 데이타를 출력한다.

그림 6은 이러한 반품을 처리하는 워크플로우를 확장된 액티비티 다이어그램으로 표현한 것이다.

그림에서 보는 바와 같이 각 참여자는 스윔레인으로 표현하였으며, 확장된 표기법인 사전조건과 사후조건, 그리고 입력 데이타와 출력 데이타를 갖는 액션노드로 표현하였다. 또한 각각의 액션노드는 자동인 경우와 수동인 경우로 구별하여 표현하였다. 자원을 나타내는 저장소 역시 확장된 표기법으로 표현하였으며, 사전조건과 사후조건과의 관계를 갖는 액션은 점선의 화살표로, 자원과 연관된 액션은 점선의 직선으로 표시하였다.

이와 같은 워크플로우 모델링의 액티비티 다이어그램 노드들 자체는 앞에서 시멘틱으로 표현되었다. 즉 노드 자체는 ASM 시멘틱으로 표현되며, 액티비티 다이어그램 역시 ASM 시멘틱으로 표현할 수 있다. 그림 6에 대

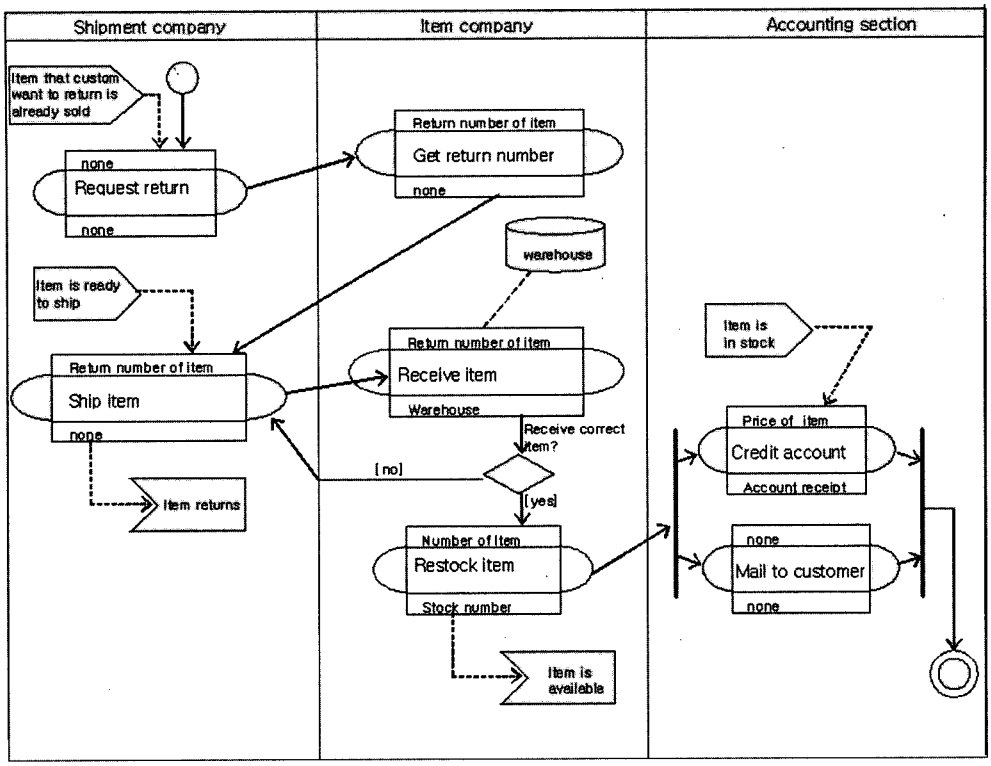


그림 6 반품처리 작업에 대한 워크플로우 모델링

하여 ASM 시멘틱으로 표현하기 위하여 다음의 보조함수를 사용한다.

makeLink(*node*, *node'*)
connect(*transition*, *node*)
connect(*node*, *transition*)
makeGraph(*P*, *in*, *out*)

*makeLink*는 노드와 노드 사이의 링크를 만들어 노드들을 연결시켜주는 함수이며, *connect*는 이러한 전이와 노드를 연결해주는 함수이다. *makeGraph*는 액티비티 다이어그램의 그래프를 만들어 주는 함수로서 액티비티 다이어그램의 스텝들을 시멘틱으로 표현한다.

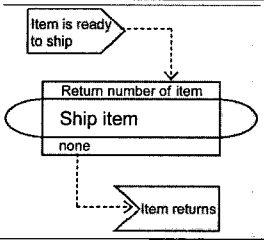
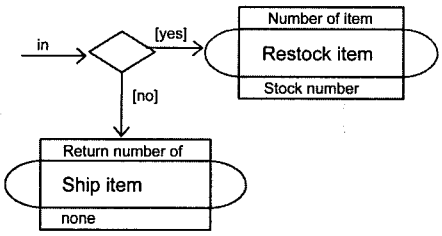
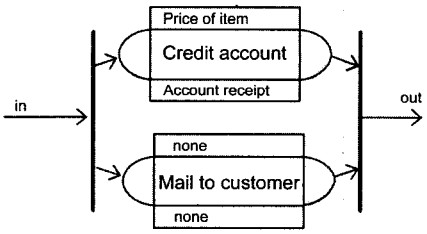
그림 6에 나타난 워크플로우 모델링 중에서 일부분만을 보조함수를 이용하여 ASM 시멘틱으로 제시한다. 다음 표 6은 그림 6을 ASM 시멘틱을 이용하여 표현한 것이다.

본 논문에서 제안한 기법은 ASM 시멘틱을 적용한 확장된 액티비티 다이어그램에 의한 워크플로우 모델링 기법이다. 기존의 Pi-calculus 및 FSP나 ASM 등의 시멘틱을 적용한 액티비티 다이어그램 기법들은 UML 액티비티 다이어그램 자체만을 시멘틱으로 표현했을 뿐 워크플로우 특성을 고려하여 확장된 모델링 기법이 아니다. 즉, 현재까지 UML 액티비티 다이어그램 자체만을 시멘틱으로 표현했을 뿐 시멘틱을 기반으로 한 액티비티 다이어그램을 확장시키거나 커스터마이징한 연구는 없었다. 단지 UML 액티비티 다이어그램 중 일부만을 시멘틱으로 표현한 연구가 있을 뿐이다[6].

본 논문에서 제안한 기법과 기존의 ASM 시멘틱을 적용한 액티비티 다이어그램의 기법을 워크플로우 특성을 고려하여 비교하면 다음과 같다.

Jablonski와 Bussler는 워크플로우 모델의 관점으로 워크플로우의 중요한 컴포넌트에 대하여 연구하였다[26]. 여기서는 모든 워크플로우 모델이 획득해야만 하는 중요한 관점으로 다섯 가지의 컴포넌트 관점을 제시한다. 먼저 기능(function) 관점은 실행될 작업 프로세스의 기능적인 단위들을 묘사하며 오퍼레이션(operation) 관점은 어떻게 워크플로우 오퍼레이션이 구현되는가를 묘사한다. 다음으로 행위(behavior) 관점은 워크플로우 제어 흐름을 묘사하며 정보(information) 관점은 워크플로우의 데이타의 흐름을 묘사한다. 마지막으로 조직(organization) 관점은 워크플로우 또는 워크플로우 애플리케이션 실행을 부가하여야 하는지에 대하여 묘사

표 6 시멘틱으로 표현된 스텝들

스텝	시 멘 틱
	$makeGraph(PREPPOST, in, out)$ $makeGraph(PRE, out)$ $makeGraph(P, in, out)$ $makeGraph(POST, in)$
	$makeGraph(\text{IF } a \text{ THEN P ELSE Q}, in, out)$ let β=new(BRANCHNODE) $yes = makeLink(\beta, a)$ $no = makeLink(\neg\beta, a)$ $connect(in, \beta)$ $mkGraph(P, yes, out)$ $mkGraph(Q, yes, out)$
	$makeGraph(P_1, P_2, in, out)$ let $\alpha = new(FORKNODE)$ $\alpha_1 = makeLink(\alpha, transition)$ $\alpha_2 = makeLink(\alpha, transition)$ $\beta = new(JOINNODE)$ $\beta_1 = new(transition)$ $\beta_2 = new(transition)$ $connect(in, \alpha)$ $makeGraph(P_1, \alpha_1, \beta_1)$ $makeGraph(P_2, \alpha_2, \beta_2)$ $connect(\beta_1, \beta)$ $connect(\beta_2, \beta)$ $connect(\beta, out)$

한다.

본 논문에서는 이와 같은 관점들 중 오퍼레이션 관점을 제외한 나머지 네 가지 컴포넌트를 기준으로 비교한다. 즉, 기능 관점, 행위 관점, 정보 관점, 조직 관점의 네 가지 관점으로 비교한다. 왜냐하면 오퍼레이션 관점은 어떻게 워크플로우 오퍼레이션이 구현되는지의 구현에 초점을 둔 관점이기 때문이다. 표 7은 이러한 관점들을 기준으로 본 논문에서 제안한 기법과 기존의 ASM을 적용한 액티비티 다이어그램 기법[16]을 비교한 것을 나타낸다.

표에서 보는 바와 같이 워크플로우 모델링이 가져야 하는 주요 관점으로 기능, 행위, 정보, 조직 관점을 기준으로 비교하였다.

• 기능 관점

먼저 동적인 측면을 놓고 비교하면 두 가지 기법 모두 동적인 측면을 지원하고 있다. 액티비티 다이어그램 자체가 시스템의 동적인 흐름을 표현하는데 있어 적합하기 때문이다. 목표 지향적 특성은 워크플로우가 갖는 주요한 특성으로 하나의 워크플로우가 종료되면 목표가 달성되어야 한다. 본 논문의 기법에서는 이러한 목표 지향적 특성의 표현이 명확하지는 않다. 기존의 기법들에서 역시 목표 지향적인 측면 지원이 부족한 것으로 보아 워크플로우 특성의 반영이 미비한 상태로 평가할 수 있다. 왜냐하면 액티비티 다이어그램이 종료되었다고 해서 목표가 달성되었다고 말할 수 없기 때문이다. 그러나 목표를 달성하기 위해서 이벤트를 수행해 나가는 과정을 표현하고 있으므로 목표 지향적인 특성이 전혀 없다고 할 수는 없다.

표 7 본 논문의 연구와 다른 기법들과의 비교

워크플로우 모델이 획득해야 하는 주요 관점		본 논문에서 제안한 기법	ASM 시멘틱을 적용한 기존의 액티비티 다이어그램 기법
기능 관점	동적인 측면	동적측면 지원	동적측면 지원
	목표 지향	액티비티 다이어그램 종료가 목표 달성은 아님	액티비티 다이어그램 종료가 목표 달성은 아님
행위 관점	조건과 제약사항	사전조건과 사후조건으로 제약사항을 표현	조건이나 제약 사항의 표현이 부족함 전이를 통해 제약사항을 표현
	시간	시멘틱으로 정의되나 노드의 표기로는 지원되지 않음	시간적인 요인의 표현이 부족
정보 관점	데이타의 흐름	입력 및 출력 데이타로 데이타의 흐름을 알 수 있음	데이타의 흐름이 지원되지 않음
	이벤트 지향	액티비티 다이어그램 자체가 이벤트 지향	액티비티 다이어그램 자체가 이벤트 지향
조직 관점	참여자	참여자는 스윔레인으로 표현	참여자는 스윔레인으로 표현
	자원	자원 노드로 사용되는 자원을 표기	사용되는 자원을 표기할 수 있는 표기법이 없음

• 행위 관점

기존의 ASM 시멘틱을 적용한 액티비티 다이어그램 기법은 조건이나 제약 사항을 전이(transition)를 통해 표현한다. 하지만 전이만을 통해서는 제약사항을 자세하게 표현할 수 없다. 그러나 본 논문에서 제안한 기법에서는 사전조건과 사후조건을 명시하여 이러한 제약사항을 명시하고 있는 장점이 있다. 또한 시간 특성을 고려해 보았을 때 본 논문에서 제안한 기법에서는 완벽하게 지원해주고 있다. 시간적인 요인은 워크플로우에서 중요한 특성 중의 하나이다. 그러나 이러한 시간적인 부분이 ASM 시멘틱을 적용한 기존의 액티비티 다이어그램 기법에서는 완벽한 지원이 이루어지지 않고 있다.

• 정보 관점

두 가지 방법 모두 이벤트 지향의 방법으로 액티비티 다이어그램에서 이벤트를 지향하고 있음을 알 수 있다. 데이타의 흐름 특성을 살펴보았을 때 본 논문에서 제안한 기법에서는 입력 데이타와 출력 데이타의 흐름을 통하여 데이타의 흐름이 지원되고 있으나 ASM 시멘틱을 적용한 기존의 액티비티 다이어그램 기법에서는 데이타의 흐름이 지원되고 있지 않다.

• 조직 관점

액티비티 다이어그램을 기반으로 하고 있는 두 가지 방법 모두에서 스윔레인을 통하여 참여자를 표현할 수 있다. 또한 자원의 표기에 있어서 본 논문에서는 자원 노드를 확장하여 지원해주고 있으나 ASM를 적용한 기존의 액티비티 다이어그램 기법에서는 자원을 표기할만한 대안이 없는 실정이다.

이와 같이 본 논문에서는 ASM 시멘틱을 적용한 액티비티 다이어그램의 확장된 기법을 제시하였다. 따라서 본 논문에서 제안한 ASM 시멘틱을 적용한 액티비티 다이어그램의 확장된 기법은 워크플로우 특성을 반영함으로써 보다 효과적이고 효율적인 워크플로우 모델링을 할 수 있도록 해준다. 또한 기존의 UML 액티비티 다이어그램을 확장하여 사용함으로써 사용자로 하여금 보다 친근한 인터페이스를 통하여 워크플로우 모델링을 가능하게 한다.

6. 결론 및 향후 연구 방향

시스템의 동적인 측면을 표현하는데 사용되는 액티비티 다이어그램을 통하여 비즈니스 프로세스와 워크플로우를 모델링하면 효율적이다. 그러나 OMG에서 제공하는 시멘틱으로는 여러가지 모호성이 야기되기 때문에 산출물을 모델링하기 위한 정확한 이해를 하는데 시간을 소비한다. 이에 본 논문에서는 액티미티 다이어그램의 시멘틱을 ASM 시멘틱을 적용하여 제시하였다. 즉, 명확하고 정확하며, 표기법의 코딩의 최소화와 함께 문제 도메인의 개념과 용어의 사용을 고려하고, 단순한 구문 사용을 통해서 읽고 쓰기가 쉬운 이해성과 실행 가능성 그리고 여러 도메인에서 유용한 보편성 등의 장점을 갖는 ASM 시멘틱을 선정하여 제시하였다. 더욱이 ASM 시멘틱 기법을 이용하여 액티비티 다이어그램의 시멘틱을 정의하고 워크플로우의 특성을 갖는 시스템에 ASM 시멘틱을 확장하여 이를 적용하였다.

액션에서 생산되거나 사용될 데이타를 표현하는 입력 데이타와 출력 데이타, 액션의 조건을 표현하는 것으로 사전 조건과 사후 조건, 활동을 수행하기 위해 요구되는 것들을 표현하는 자원, 작업을 수행하는 자원을 가리키는 참여자, 그리고 수동 액션 또는 자동 액션의 특성을 갖는 액션 등의 워크플로우 특성을 고려하여 확장하였다. 또한 액션에서 생산되거나 사용될 입력 데이타와 출력 데이타, 수동액션인지, 자동액션인지를 나타내는 액션의 상태를 나타내는 정보들을 액션노드에서 확장하였다. 아울러 액션 상태에 의존하는 사전조건과 사후조건들을 확장하고, 확장된 ASM 시멘틱을 액티비티 다이어그램에 적용시켰다.

본 논문에서 제시하는 ASM 시멘틱을 기반으로 하는

액티비티 다이어그램의 확장된 개념으로 워크플로우를 모델링하게 되면 정형적이고 공통적으로 정의된 시멘틱의 명세가 부족하게 됨으로써 나타나는 모호성을 없앨 수 있으며, 시멘틱 명세의 부족으로 인해 산출물을 모델링하기 위한 시간 소비를 줄일 수 있다. 즉 ASM 시멘틱을 기반으로 하는 정형적인 시멘틱의 정확한 정의를 함으로써 보다 효율적으로 워크플로우를 모델링할 수 있게 된다. 또한 본 논문의 방법을 사용하면 반드시 전문가가 아니더라도 이해할 수 있고 작업하기 쉬운 보편성을 갖고 있어 더욱 용이하고 유용하게 사용될 수 있을 뿐만 아니라, 워크플로우 수행 중에 발생 가능한 예외사항과 다양한 변화에 대처가능하게 된다. 향후 연구로는 본 연구에서 제시된 ASM 시멘틱을 기반으로 하는 확장된 액티비티 다이어그램의 표기법을 사용하여 워크플로우를 모델링할 수 있는 도구를 구현하는 연구가 필요하다.

참 고 문 헌

[1] Object Management Group, "Unified Modeling Language Specification 1.4," 2001.

[2] E. Chang, E. Gautama, and T. S. Dillon, "Extended Activity Diagrams for Adaptive Workflow Modeling," IEEE 2001 Fourth International Symposium on Object-Oriented Real-Time Distributed Computing, pp. 413-418, 2001.

[3] Weber, Ehrig, and Reisig, "Petri Net Technology," Petri Net Technologies for Modelling Communication Based Systems, Vol. 2, pp. 93-104, 2001.

[4] W. M. P. van der Aalst, "The Application of Petri Nets to Workflow Management," The Journal of Circuits, Systems and Computers, Vol. 8(1), pp. 21-66, 1998.

[5] R. Eshuis and R. Wieringa, "A Comparison of Petri Net and Activity Diagram Variants," In Proceedings 2nd International Colloquium on Petri Net Technologies for Modeling Communication Based Systems, Berlin, Germany, 2001.

[6] R. Eshuis and R. Wieringa, "Verification Support for Workflow Design with UML Activity Graphs," 24th International Conference on Software Engineering, pp. 166-176, 2002.

[7] Schader, M., and Korthaus, A., "Modeling Business Process as Part of the BOOSTER Approach to Business Object Oriented System Development Based on UML," Second International Enterprise Distributed Object Computing Workshop Proceedings, pp. 56-67, 1998.

[8] J. Rumbaugh, G. Booch, and Ivar Jacobson, The Unified Modeling Language User Guide, Addision-Wesly, 1999.

[9] Ricardo M. Bastos, Duncan Dubugras A. Ruiz, "Extending UML Activity Diagram for Workflow Modeling in Production Systems," Proceedings of the 35th Hawaii International Conference on System Sciences, pp. 123-134, 2002.

[10] Roberto W. S. Rodrigues, "Formalising UML Activity Diagrams using Finite State Processes," UML 2000 Workshop, 2000.

[11] 류재광, 김광훈, "실시간 협업지원 그룹 ICN 에디터의 설계 및 구현", 한국인터넷정보학회, Vol. 2(5), 2001.

[12] Yang Dong and Zhang ShenSheg, "Using Pi-calculus to Formalize UML Activity Diagram for Business Process Modeling," 10th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems (ECBS'03), pp. 47-54, 2003.

[13] R. Eshuis and R. Wieringa, "An Execution Algorithm for UML Activity Graphs," Proc. <<UML>> 2001, pp. 47-61, 2001.

[14] Jan Hendrik Hausmann, Reiko Heckel, and Stefan Sauer, "Toward Dynamic Meta Modeling of UML Extensions: An Extensible Semantics for UML Sequence Diagrams," IEEE 2001, pp. 80-87, 2001.

[15] Martin Gogolla and Mark Richters, "Expressing UML Class Diagrams Properties with OCL," LNCS, pp. 85-114, 2002.

[16] Y. Burevich, Specification and Validation Methods, Oxford University Press, 1995.

[17] Wuwei Shen, Kevin Compton, and James K. Huggins, "A Toolset for Supporting UML Static and Dynamic Model Checking," 26th International Computer Software and Applications Conference(COMPSAC 2002), IEEE Computer Society, Vol. 3, pp. 147-152, 2002.

[18] E. Borger, A. Cavarra, and E. Riccobene, "An ASM Semantics for UML Activity Diagram," AMAST 2000, pp. 292-308, 2000.

[19] Hammer, D. K., Hanish, A. A., and Dillon, T. S., "Modeling Behavior and Dependability of Object-Oriented Real-time Systems," Journal of Computer Systems Science and Engineering, Vol. 13(3), pp. 139-150, 1998.

[20] 원재강, 김학성, 이문영, 김광훈, 정관희, "워크플로우 표준화 동향 분석", 한국인터넷정보학회, Vol. 1(1), 2001.

[21] Haake, J. M., and Wang, W., "Flexible Support for Business Processes: Extending Cooperative Hypermedia with Process Supports," In Proceedings of the International ACM SIGGROUP-GROUP 97, pp. 341-350, 1997.

[22] Clarence A. Ellis and Gary J. Nutt, "Office Information Systems and Computer Science," Computing Surveys, Vol. 12, No. 1, 1980.

[23] Ortner W., and Starty C. "Virtualization of Organizations: Consequences for Workflow Modeling," Proceedings of the 32nd Hawaii International Conference on System Sciences, 1999.

[24] Stary C., "Integrating Workflow Representation

into User Interface Design Representation," Software Concepts and Tools Springer, 1997.
[25] 김광훈, 워크플로우 기술 I, TTA 저널, Vol. 85, pp. 107-111, 2001.
[26] Jablonski, S., Bussler, C., "Workflow Management-Modeling Concepts, Architecture and Implementation," Int. Thomson Publishing, London, 1996.

고 은 정
2002년 전북대학교 컴퓨터과학과 졸업(이학사). 2004년 전북대학교 컴퓨터과학과 대학원 졸업(이학석사). 관심분야는 시멘틱 워크플로우, 소프트웨어공학, CRM 등임

이 상 영
1994년 숭실대학교 산업공학과 졸업(공학사). 1998년 전북대학교 대학원 산업공학과 졸업(공학석사). 2004년 전북대학교 대학원 전산통계학과 졸업(이학박사). 2005년~현재 남서울대학교 보건행정학과 전임강사. 관심분야는 의료정보, 워크플로우, CRM, 모바일 애플리케이션 등임

유 철 중
1982년 전북대학교 전산통계학과 졸업(이학사). 1985년 전남대학교 대학원 계산통계학과 졸업(이학석사). 1994년 전북대학교 대학원 전산통계학과 졸업(이학박사). 1982년~1985년 전북대학교 전자계산소 조교. 1985년~1996년 전주기전여자대학 전자계산과 전임강사~부교수. 1997년~현재 전북대학교 자연과학대학 컴퓨터과학과 전임강사~부교수. 관심분야는 소프트웨어 개발 프로세스, 소프트웨어 품질, 컴포넌트 소프트웨어, 소프트웨어 메트릭스, 소프트웨어 에이전트, GNSS, GIS, 교육공학, 인지과학 등임

장 옥 배
1973년 고려대학교 졸업(학사,석사). 1988년 산타바바라대 대학원(Ph. D.) 1974년~1980년 죠지아 주립대. 오하이오 주립대 박사과정 수료. 1980년~현재 전북대학교 자연과학대학 전자정보공학부 교수. 관심분야는 소프트웨어공학, 전산교육, 수치해석, 인공지능 등임