지식레벨의 전문가시스템 구축을 위한 태스크 객체 모델링 기법

김 은 경*, 김 성 훈**, 박 충 식**
한국기술교육대학교 컴퓨터공학과* 영동대학교 전자공학부**

Task Object Modelling Technique for Knowledge Level Expert System Implementation

Eun-Gyung Kim*, Seong-Hoon Kim**, Choong-Shik Park**
Dept. of Computer Eng., Korea University of Tech. & Edu.*
Faculty of Electronic Eng., Youngdong University**

요약

최근 지식레벨에서 전문가시스템을 개발함으로써 심볼레벨에서 개발된 전문가시스템의 문제점을 해결하기 위한 여러 가지 방법들이 제안되었다. 그러나, 대개 개념적인 모델링 기법을 제시하는 수준에 머무르고 있다. 본 논문에서는 이러한 개념적인 모델링 기법이 실용 가능한 개발기법이 될 수 있도록, 기존의 연구 내용을 수정, 보완한 태스크 객체 모델링(TOM) 기법을 제시하였다. 이 기법에서 태스크 객체(TO)는 자신의 목표와 수행 조건, 실행할 행위 지식 등을 포함하는 하나의 지식 단위로 정의하고, TO의 구조적, 동적, 기능적 측면을 쉽게 도식화할 수 있는 태스크 객체 다이어그램(TOD)을 정의하였다. 또한, 각 TO의 행위 지식을 표현하는 기본 단위로서 추론유형(Inference Type)들을 정의하고, TO의 상태 변화에 기반을 둔 TO 처리 알고리즘을 개발하였으며, 이를 기초로 TO의 상태를 모니터링할 수 있는 태스크 디버거를 구현하였다.

1. 서론

전문가시스템은 단순히 전문가로부터 획득된 지식을 보관하는 데이터베이스가 아니라, 실세계의 현상으로 묘사될 수 있는 어떤 적절한 행위를 나타내는 조작 모델(operational model)이라고 볼 수 있다. 따라서 전문가시스템의 개발과정을 모델링 행위로 보는 것이 바람직하며, 지식 공학자의 가장 중요한 역할은 문제분야의 지식을 습득하여 문제해결에 적합한 시스템 모델을 개발하는 것이라고 할 수 있다. 따라서 전문가의 지식표현 체계와 일치하면서 지식 공학자의 모델링 행위가 자연스럽게 시스템 개발과 연결될 수 있는 모델링 기법이 필요하다. 한편, 기존의 지식표현 방법은 문제풀이 지식과 그 지식에 대한 처리가 분리되어 있는 심볼레벨(symbol level)의 표현 체계를 가지므로 전문가의 복합적인 지식체계와 사고과정을 기술하기

에 적절하지 않으며, 결과적으로 지식습득이 전문가시스템 개발시에 병목 현상을 유발하는 주요 원인이 되고 있다(Waterman, 1986). 지식레벨(knowledge level)이란 용어는 Newell이 자신의 논문에서 가장 먼저 사용한 것으로, 심볼레벨 위에서 기존의 지식표현 방법으로 기술된 독자적인 목표와 문제풀이 방법을 포함하고 있는 지식의 한 단위라고 정의할 수 있다(Newell, 1992)(박충식, 1992)(박충식, 김재희, 1992).

본 논문에서는 지식레벨에서의 전문가시스템 개발의 필요성을 인식하고, 현재 가장 널리 사용되고 있는 전문가시스템 개발도구 가운데 하나인 뉴론 데이터(Neuron Data)의 IRE(Intelligent Rules Element)(NeuronData, 1995) 상에서 지식레벨의 전문가시스템 개발 기법을 연구하였다. 먼저 성취할 목표와 그 목표를 성취하는데 필요한 지식 및 처리기능을 "태스크 객체(Task Object: TO)"라는 하나의 지식 단위로 정의하고, TO의 구조적, 동적, 기능적 측면을 쉽게 도식화할 수 있는 "태스크 객체 다이어그램(Task Object Diagram: TOD)"을 정의함으로써 인간의 작업 단위와 마찬가지로 지식레벨에서 전문가시스템을 개발할 수 있는 "태스크 객체 모델링(Task Object Modelling: TOM)" 기법을 제안하였다.

한편, KADS(Schreiber, 1994)(KADSII, 1998)나 SKE(Structured Knowldege Engineering)(Gardner, 1991)(Bechel, 1990) 등과 같은 기존의 연구에서도 지식레벨에서의 전문가시스템 개발기법들이 제안되었으나, 대부분 개념적인 수준의 기법들이므로 실제로 전문가시스템을 개발할 때 상용화된 전문가시스템 개발도구에서 적용하기에는 많은 어려움이 있다. 그러나 본 논문에서는 TO의 상태 변화에 기반을 둔 TO 처리 알고리즘을 개발하고, 이를 기반으로 태스크 디버거를 개발함으로써 TOM 기법을 기반으로 개발된 전문가시스템을 구성하는 TO들의 상태를 모니터링할 수 있도록 지원한다. 또한, 각 TO의 행위 지식을 표현하는 기본 단위로서 추론유형(Inference Type)들을 정의하였다. 마지막으로 TOM 기법을 적용하여 통신망 설계 전문가시스템의 프로토타입 시스템을 개발하고, TOM 기법의 실용성을 평가하였다.

2. 태스크 객체 모델링 기법 개요

2.1 태스크 객체 모델링의 개념

기존의 지식레벨 개발기법은 대부분 문제분야 계층(domain layer)과 추론 계층, 태스크 계층, 전략 계층(strategy layer)의 4층 구조로로 이루어져 있다. 문제분야 계층은 문제분야의 개념과 개념의 속성, 개념들간의 관계 등을 기술하는 부분이며, 추론 계층에는 해석모델(interpretation model)을 이용하여 구체적인 추론 내용을 기술한다. 태스크 계층에는 태스크 단위의 문제풀이 업무을 상세히 기술하며, 이는 추론제어 지식에 해당한다. 전략 계층에서는 문제풀이 업무의 순서 결정과 같은 문제풀이 전략을 기술하게 된다. 심볼레벨의 전문가시스템 개발도구는 태스크 계층과

추론 계층이 혼합되어 있는 형태로, 지식습득 및 지식베이스 구축 과정에서 추론내용 지식과 추론제어 지식을 분리시키는 것이 어려우며, 결과적으로 전문가시스템 개발을 매우 어렵게 한다. 따라서 추론내용 지식에 해당되는 추론 계층과 추론제어 지식에 해당되는 태스크 계층이 명확히 분리되는 지식레벨의 개발기법이 절실히 요구된다. 본 논문에서 제안한 태스크 객체 모델링(TOM) 기법에서는 태스크 계층을 TOD로 표현하고, 전략 계층은 별도로 설정하지 않고 TOD에서 AND/OR 그래프 형태로 표현되도록 함으로써 그림 1과 같이 3층 구조로 구성하였다. 추론 계층에서는 고정된 해석모델을 제시하는 대신 기본적인 추론 유형만을 정의하였다. 대표적인 지식레벨 개발기법인 SKE와 KADS에서는 모니터링, 설계, 예측 등과 같은 몇가지 고정된 해석모델을 제시하고 있으며, 시스템 개발시 해결하려는 문제에 적합한 한가지 이상의 해석모델을 선택하여 통합, 수정해야 하므로 오히려 시스템 개발이 어렵고 융통성도 떨어지게 된다. 한편, 문제분야 계층에서는 IRE의 클래스와 객체 개념을 이용하여 문제분야의 지식을 표현한다.

태스크 계층	TOD 및 TO 정의
추론 계층	추론유형 정의
문제분야 계층	문제분야 지식 표현

(그림 1) TOM 기법의 3층 구조

2.2 태스크 객체의 정의

태스크 객체(TO)는 전문가가 수행하는 일의 단위, 또는 지식 공학자가 판단한 일의 단위가 될 수 있다. 기존의 지식표현 방법은 지식표현과 그 지식을 처리하는 기능이 완전히 분리되어 있기 때문에 이러한 지식이 종합되어 있는 인간의 지식을 자연스럽게 표현하기가 매우 어렵다. 따라서 TO는 인간의 작업 단위와 일치하면서 독자적인 목표와 그 목표를 성취하는데 필요한 지식 및 처리 기능이 하나의 단위가 되도록 정의되어야 한다. TO는 전체 시스템을 표현하는 TOD의 구성 요소이므로, 전체 시스템내에서 각 TO가 실행될 시기, 실행에 필요한 조건, 구체적인 실행 내용 등이 표현될 수 있어야 한다. 또, TO의 조합으로 전문가시스템을 모델링하기 위해서는 TO의 구조적, 동적, 기능적인 측면을 정의할 수 있어야 한다. 본 논문에서는 이러한 요구 사항들을 종합하여 그림 2와 같이 Task라는 클래스를 정의하였다. 구조적인 측면은 한 TO를 구성하는 부태스크 객체(SubTask Object: STO)들의 구성 방법을 의미하며, TO를 정의할 때 그림 2의 CompositionType과 ResultType에 기술된 정보가 종합되어 STO의 구성 방법이 정해진다. 즉, 한 TO의 모든 STO의 상태가 성공(success)이 되어야 하는 경우, CompositionType은 'AND'로, ResultType은 'SUCCESS'로 정의해야 한다. 동적인 측면은 활성(ACTIVE) 상태가 될 TO로

선정되는 조건을 의미하며, STO들 사이의 수평적인 활성화 관계를 기술하는 것으로, SuccessEventList와 FailEventList에 기술한다. 기능적인 측면은 DataTest와 TaskBody에 기술되며, DataTest는 TO 처리에 필요한 자료를 확인하는 항목이며, TO가 활성화되었을 때 목표를 성취하기 위해 실제로 수행할 내용은 TaskBody 부분에 기술한다.

(그림 2) Task 클래스의 정의

■ Name : TO를 구분할 수 있도록 이름을 기술한다.

■ Subtasks : TO를 구성하는 STO들의 이름을 기술한다.

■ ResultType : CompositionType과 종합되어 STO들의 구성 방법이 정해지며, 다음과 같은 값을 가질 수 있다.

① SUCCESS : STO의 상태가 성공이 되어야 함

② FAIL : STO의 상태가 실패가 되어야 함

③ DONE : STO의 상태와 상관없이 모든 부태스크들을 수행해야 함

■ CompositionType

① AND : 모든 STO가 ResultType을 만족해야 한다.

② OR : STO 가운데 하나만 ResultType을 만족하면 된다.

단, ResultType이 'DONE'인 경우에는 선택할 필요가 없다.

■ SuccessEventList : "성공" 상태일 때 자신을 활성화시킬 수 있는 TO들의 이름을 기술한다.

■ FailEventList : "실패" 상태일 때 자신을 활성화시킬 수 있는 TO들의 이름을 기술한다.

■ DataTest : TO 처리에 필요한 자료값을 기술한다.
■ State : TO의 상태를 나타내며, 다음과 같은 값을 가질 수 있다.
① ACTIVE : TO가 실행될 수 있는 상태
② WAIT : STO들의 실행이 완료되기를 기다리는 상태
③ SUCCESS : 수행 결과 TO의 목표가 성공적으로 성취된 상태
④ FAIL : 수행 결과 TO의 목표가 성공적으로 성취되지 못한 상태
⑤ INACTIVE : TO의 상태가 정해지지 않은 상태

각 TO가 생성되는 초기상태에서는 TOD의 루트 TO의 상태만 'active'이고, 나머지 TO는 'UNKNOWN' 상태이다.

■ Cycle : TO를 1회만 수행할 지 또는 조건이 만족될 때까지 반복 수행할 지를 기술하며, TRUE나 FALSE로 구분한다.

■ RepeatCondition : TO의 Repeat 항목이 TRUE인 경우, 반복이 종료될 조건을 TO의 상태, 즉 'SUCCESS'나 'FAIL'로 기술한다.

■ TaskBody : TO가 활성화되었을 때 수행할 행위를 기술하는 곳으로, 하나 이상의 IRE 규칙의 가설(hypothesis) 이름을 나열한다.

2.3 태스크 객체 다이어그램의 정의

본 논문에서 제안한 TOM 기법에서는 전체 시스템이 TO들에 의해서 계층 구조로 모델링되는데, 전문가시스템의 최종 목표를 갖고 있는 루트 TO와 여러개의 STO들로 구성된다. 본 논문에서는 이러한 계층적인 구조를 그림으로 쉽게 표현할 수 있도록 태스크 객체 다이어그램(Task Object Diagram: TOD)을 정의하였다. 이는 소프트웨어 공학에서 사용되는 객체지향 개발기법의 다이어그램을 약간 변형한 것으로, TOD는 크게 다음 2가지 요소로 구성된다.

(1) 노드(node): TO의 구조적 측면을 표현할 수 있도록 다음과 같이 구분하였다.
① CompositionType AND/OR : 육각형과 원으로 구분한다.
② ResultType SUCCESS/FAIL/DONE : 노드 안에 각각 S, F, D로 구분한다.
③ Cycle TRUE/FALSE : 반복 화살표의 유무로 구분한다.

(2) 링크(link): TO와 STO들간의 관계뿐만 아니라, TO의 동적인 측면을 표현할 수 있도록 다음과 같이 구분하였다.
① 실선 화살표 : TO와 STO들간의 관계 표시
② 점선 화살표 : 성공 이벤트 관계 표시
③ 굵은선 화살표 : 실패 이벤트 관계 표시

2.4 태스크 객체 처리 알고리즘

TOD의 루트 TO에 명시된 최종 목표를 성취하기 위해서는, TO의 구조적인 측면과 동적인 측면을 고려하여 각 TO의 상태를 적절히 변경하면서 전체 TO를 실행해야 한다. 또한, 각 TO가 활성화되면 자신의 상태가 성공 또는 실패로 판단될 때까

```
Procedure  Task_execute()
 Begin
   task : active 상태인 태스크 이름 /* task.State가 "active"인 task선택 */
   Success_event_check(task);
     /* SuccessEventList에 기술된 task들의 State가 "success"인지 체크 */
   Fail_event_cist(task);
     /* FailEventList에 기술된 task들의 State가 "fail"인지 체크 */
   If  task.subtasks = NULL  /* subtask가 없는 경우 자신을 처리 */
    Then  Run_taskbody(task); /* task의 TaskBody 부분 실행 */
          State_modify(task);  /* task의 상태 수정 */
          Repeat_check(task);  /* task의 반복 실행 여부 판단 */
    Else   task.state := "wait";  /* task의 상태를 "wait"으로 변경 */
      Case  task.ResultType  Is  /* ResultType에 따라 처리 구분 */
        [DONE] :
          모든 subtask들의 상태가 "unknown"이 아닐 때까지
          Task_execute()와 Repeat_check()를 반복 수행한다.
        [SUCCESS] :
          Case  task.CompositionType Is
            [OR] : subtask 중 하나의 State가 "success" 이거나
                   모든 task의 State가 "unknown"이 아닐 때까지
                   Task_execute()와 Repeat_check()를 반복 수행한다.
            [AND] subtask 중 하나의 State가 "fail"이거나
                   모든 task의 State가 "unknown"이 아닐 때까지
                   Task_execute()와 Repeat_check()를 반복 수행한다.
          EndCase
        [FAIL] :
          Case task.CompositionType Is
            [OR] : subtask 중 하나의 State가 "fail"이거나
                   모든 task의 State가 "unknown"이 아닐 때까지
                   Task_execute()와 Repeat_check()를 반복 수행한다.
            [AND] subtask 중 하나의 State가 "success"이거나
                   모든 task의 State가 "unknown"이 아닐 때까지
                   Task_execute()와 Repeat_check()를 반복 수행한다.
          EndCase
      EndCase
      State_check(task);  /* subtask들의 상태나 자신의 TaskBody 부분의
                             실행 결과에 따라 task의 State를 변경한다. */
 End Task_execute;

/*  Main  module  */
Begin
   Root := active 상태인 태스크의 이름;  /* TOD의 root task */
   Task_execute();   /* 태스크 실행 */
   Repeat_check(Root);   /* 반복 수행 여부 판단 */
End;
```

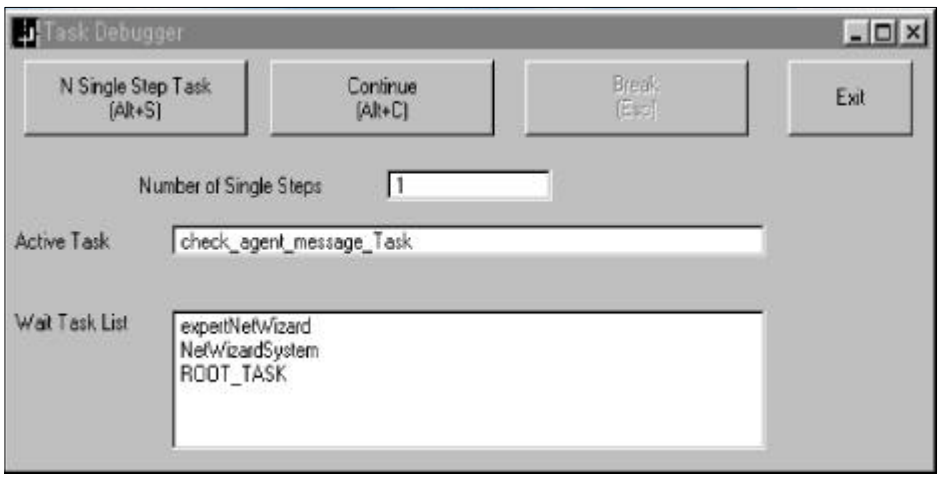
(그림 3) TO 처리 알고리즘

지 자신의 목표를 성취하는데 필요한 행위를 1회 또는 반복 수행해야 한다. 본 논문에서는 각 TO의 상태를 기반으로 하는 TO 처리 알고리즘을 개발하고, 이를 기

반으로 태스크 객체 디버거를 IRE의 API 함수로 구현하였다. 그림 3은 TO 처리 알고리즘을 개략적으로 표현한 것이다.

2.5 태스크 디버거

그림 4는 태스크 디버거의 인터페이스를 표시한 것이다. TOM 기법을 기초로 IRE를 이용하여 전문가시스템을 구현한 경우, IRE 자체에는 TO라는 개념이 없으므로 TO들이 제대로 수행되고 있는지 확인할 방법이 없다. 그러나 이 디버거를 이용하면 전문가시스템을 구성하는 TO들의 실행 상태를 모니터링할 수 있으며, 결과적으로 개발자의 의도대로 TO들이 수행되고 있는 지를 쉽게 확인할 수 있도록 지원한다.



(그림 4) 태스크 디버거

2.6 추론유형 정의

추론유형(inference type)이란 추론 단계에서 하나의 행위를 수행하는 최소 단위로, 어떤 입력 데이터에 대해서 행위를 수행한 결과로 새로운 지식 또는 정보를 출력하게 된다. 즉, 특정 개발도구의 지식표현 방법과는 무관한 각 TO의 행위 지식을 표현하는 기본 단위이다. 본 논문에서는 SKE의 추론유형을 참조하여 표 1과 같이 추론유형을 정의하였으며, 필요한 경우 새로운 추론유형들이 계속 추가될 수 있다. 추론유형의 정의에 사용된 용어는 다음과 같이 정의한다. 개념(concept)은 문제분야 지식 가운데 핵심이 되는 객체(object)로서 이름으로 구분되며, 하나 이상의 속성(property)을 갖는다. 속성은 이름과 속성이 가질 수 있는 값(value)으로 정의된다. 구조(structure)란 하나 이상의 개념이나 그들간의 관계로 구성되는 복잡한 객체를 의미하며, 전체 시스템을 하나의 구조로 볼 수 있다.

<표 1> 추론유형의 정의

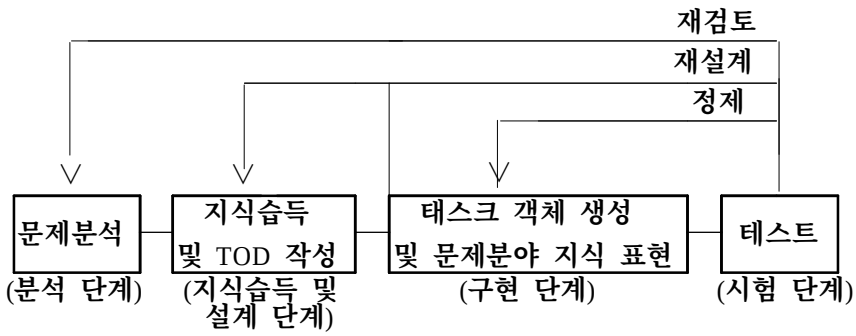
연산 유형	추론 유형	인수(arguments)
Change Concept	assign	property -> property-value
	compute	structure -> property-value
Generate New Concept	instantiate	concept -> instance
	identify	instance -> concept
	generalize	set of instances -> concept
	abstract	concept -> concept
	specify	concept -> concept
	select	set of concepts -> concept
	heuristic-match	concept -> concept
Similarities /Differences	compare	value + value -> value
	match	structure + structure -> structure
	confirm	value -> value
Structure Manipulation	compose	set of instances -> structure
	decompose	structure -> set of instances
	transform	structure -> structure

2.7 문제분야 지식

문제분야 지식은 전문가시스템을 개발하려는 분야의 전문지식(expertise)을 의미하며, 각 TO가 수행할 몸체(TaskBody) 부분에서 실행될 규칙을 작성할 때 나타나는 자료에 해당한다. 이 지식의 표현은 실제로 사용할 전문가시스템 개발도구에 따라 달라지며, IRE를 사용하는 경우 클래스와 객체, 프로퍼티 등으로 문제분야 지식을 표현하게 된다. 이 부분에 대해서는 3장에서 예를 보이겠다.

2.8 TOM 기법에 기반을 둔 전문가시스템 개발 단계

TOM 기법을 기반으로 전문가시스템을 개발하는 단계를 도식화하면 그림 5와 같다. TOM 기법에서 각 TO는 지식공학자의 업무처리 단위와 일치하므로 지식습득 단계에서 TOD를 작성하는 것이 매우 용이하며, 결과적으로 지식습득 단계에서의 병목현상을 최소화할 수 있다.



(그림 5) TOM 기법을 이용한 전문가시스템의 개발 단계

3. 적용 사례: 통신망 설계 전문가시스템의 프로토타입

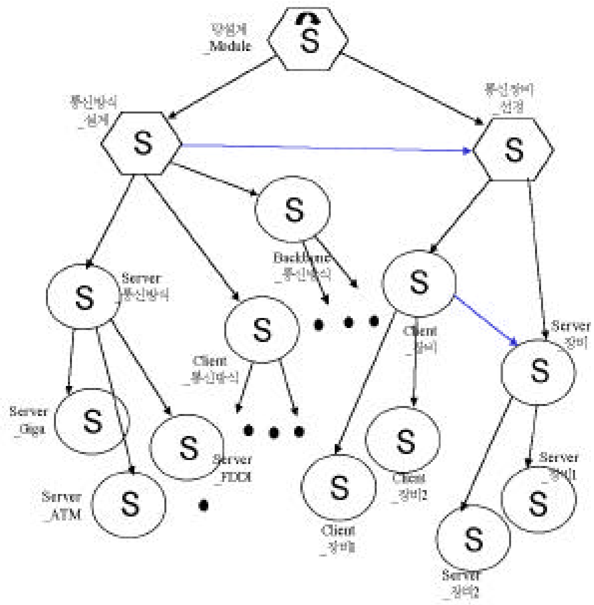
본 논문에서 제안한 TOM 기법의 적용 사례로서 통신망 설계 전문가시스템의 전체 기능 가운데 통신 방식과 통신 장비를 선정하는 부분만을 선택하여 프로토타입 시스템을 개발하였다.

3.1 TOD 작성

통신망 설계 전문가로부터 습득한 지식으로 모델링된 TOD는 그림 6과 같다. 프로토타입에서 망설계_Module TO는 통신방식_설계와 통신장비_선정이라는 두개의 STO로 구성된다. 또한, 통신방식_설계 TO가 성공적으로 수행된 다음에만 통신장비_선정 TO가 수행될 수 있으므로, 통신방식_설계와 통신장비_선정 TO 사이에 저선 화살표를 사용하여 성공 이벤트 관계를 표현하였다. 통신방식_설계 TO는 Server와 Client, Backbone의 통신방식을 선정하는 세 개의 STO로 구성되며, 이들 간의 수행 순서는 중요하지 않지만 모두 성공적으로 수행되어야 통신방식_설계 TO가 실행될 수 있으므로, SUCCESS와 AND를 의미하는 S-육각형 노드로 표시하였다. Server_통신방식 노드는 여러개의 말단 TO들로 구성되며, 이들 가운데 하나만 성공 상태가 되면 Server_통신방식 TO가 실행될 수 있으므로, SUCCESS와 OR를 의미하는 S-원 노드로 표시하였다. 또한, 망설계_Module TO는 성공 상태가 될 때까지 반복 수행되어야 하므로, 노드에 반복 화살표를 표시하였다.

3.2 TO 생성

그림 7은 그림 6의 TOD 가운데 하나인 Server_Giga 노드의 TO를 생성한 예이며, 이 노드의 TaskBody에 기술된 가설 이름을 갖는 IRE의 규칙까지 함께 보여준다.



(그림 6) 통신망 설계 전문가시스템의 TOD

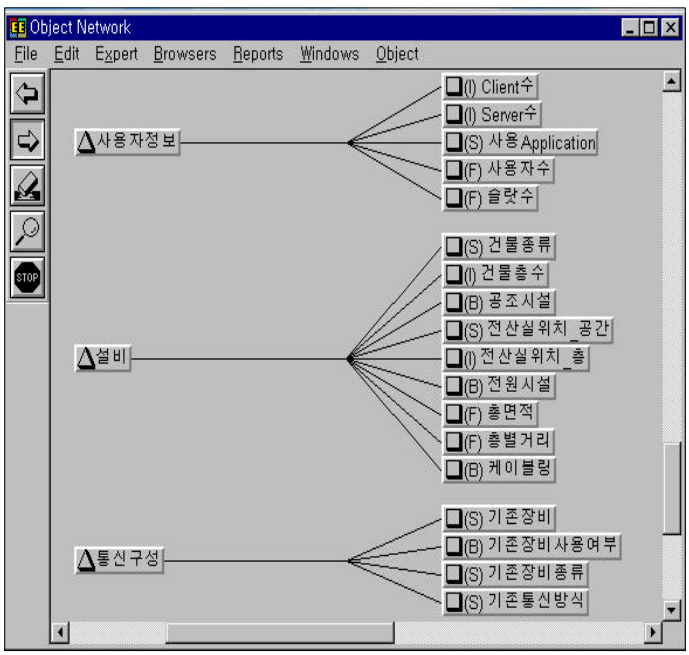
```
Task   Server_Giga
 Classes   Task
 Properties
  Name: Giga
  Subtasks:
  CompositionType:
  ResultType: DONE
  SuccessEventList:
  FailEventList:
  DataTest:
  ActionGoal:
  State: unknown
  Cycle: FALSE
  RepeatCondition:
  TaskBody: 통신방식_Server

(@RULE=     통신방식_Server_3
  (@LHS=
    (= (논리적요소.고속성_Server) ("Dedicated 100M 이상"))
    (= (논리적요소.경제성) ("미고려")
  )
  (@HYPO= 통신방식_Server)
  (@RHS=
    (Assign ("Giga Base")(통신방식.Server방식))
    (Assign (7) (통신방식.Server방식_우선순위))
    (Assign (통신방식.Server방식) (Why_통신방식.Server))
  )
```

(그림 7) Server_Giga 노드의 TO

3.3 문제분야 지식

그림 8은 통신망 설계와 관련된 전문지식을 표현하기 위해서 IRE에서 정의한 객체의 일부와 그 프로퍼티들의 관계를 보여주는 IRE의 객체 네트워크(object network)이다.



(그림 8) 문제분야 지식의 객체 네트워크

4. 결론

본 논문에서는 전문가의 지식표현 체계와 일치하면서 지식 공학자의 모델링 행위가 자연스럽게 시스템 개발과 연결될 수 있는 태스크 객체 모델링 기법을 제안하였다. 또한, 대표적인 전문가시스템 개발도구인 IRE를 이용하여 전문가시스템을 개발할 때 실제로 이 기법을 적용할 수 있도록, TO 처리 알고리즘을 개발하고 이를 기초로 태스크 디버거를 구축하였다. 본 논문에서 제안한 TOM 기법의 실용성을 평가하기 위하여 TOM 기법에 이용하여 통신망 설계 전문가시스템을 구현해 본 결과, 지식 습득시 작성한 TOD가 바로 시스템 설계서가 될 수 있으므로 지식 습득과 시스템 설계가 분리되었던 기존의 개발 방법에 비해서 전체 과정을 크게 단축할 수 있음을 알 수 있었다.

앞으로 TOD와 TO를 직접 편집할 수 있도록 에디터를 구현하고, 추론유형과 1:1로 대응하는 IRE 메소드를 정의하여 라이브러리로 구축함으로써 TOM 기법을 보다 효율적으로 적용할 수 있도록 지원할 계획이며, TO에 기반을 둔 설명 기능 및 TO 단위의 테스팅 기법에 대해서 연구할 계획이다.

참고 문헌

1. **박충식, 태스크 개념을 이용한 전문가시스템 개발기법, 박사학위논문**, 1992.
2. **박충식, 김재희, "태스크 개념을 이용한 전문가시스템 개발기법," 지능정보시스템 제1권 제1호**, 1992.8., pp.33-48.
3. Gardner, K., "Designing Knowledge Systems with Objects," AI Expert, Sept1991, pp.32-39.
4. Newell, A., "The Knowledge Level," Artificial Intelligence 18, 1992, pp.87-127.
5. Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. and Lorensen, W., Object-Oirented Modelling and Design, Prentice-Hall, 1991.
6. Schreiber, G., et al, CommonKADS: A Comprehensive Methodology for KBS Development, IEEE Expert, December 1994, pp.28-37.
7. Waterman, D. A., A Guide to Expert System, Addison-Wesley, 1986.
8. Bechtel AI institute and BOLESIAN, Structured Knowledge Engineering Tutorial, 1990.
9. KADSII Technical Reports, SWI Home Page, http://swi.psy.uva.nl/swi.html.
10. NeuronData, IRE(Intelligent Rule Element) manual, 1995.