객체지향 페트리 넷을 이용한 계층적인 요구사항의 명세 및 검증

(Hierarchical Specification and Verification of Requirements using an Object-Oriented Petri Net)

홍 장 의 † 윤 일 철 †† 배 두 환 †††

(Jang Eui Hong) (Il-Cheol Yoon) (Doo-Hwan Bae)

요 약 요구사항이 복잡하고 다양해지면서 정형적인 방법을 이용한 시스템 명세가 방대해지고 이해하기 어려워진다는 문제들이 생겨났다. 따라서 요구사항의 명세를 위해 모듈화 및 객체화 개념 등을 도입하고 있으며, 특히 복잡한 시스템의 경우에 있어서는 요구사항을 하향식 접근 방법에 의해 분할하고, 이들을 각각 정형적으로 명세하여 합성하는 접근 방법을 사용하고 있다. 본 연구에서는 이러한 추세에 따라 요구사항을 계층적으로 모델링하고, 객체지향 개념을 충분히 표현할 수 있는 정형적인 방법을 제안한다. 제안된 객체지향 페트리 넷인 HOONet은 모듈화, 객체화, 추상화, 및 상세화 등의 개념을 지원하도록 설계되었으며, 복잡한 요구사항을 체계적으로 명세할 수 있는 방법을 제공한다. 특히 요구사항이 부분적으로 제시되었거나, 분석이 전체적으로 완료되지 않은 상황에서도 명세 및 검증이 가능하도록 하였으며, 점진적인 명세의 합성을 통해 시스템 모델링이 이루어지도록 하였다.

Abstract As the requirements of a software system become large and complex, it causes some problems such that requirements specification using formal methods becomes larger in its size and less understandable. In order to solve such problems, the concepts of modularity and object are adopted to specify the requirements. In addition, top-down and compositional approach to handle such requirements are also adopted. In our paper, we suggest an object-oriented Petri net, called HOONet, to hierarchically specify and verify the complex requirements by incorporating the concepts of modularity, object, abstraction and refinement into a formal method. Our HOONet method supports the incremental specification and verification of partially described or not yet fully analyzed requirements. We also show the applicability of our method by modeling and verifying the requirements of a reactor safety control system.

1. 서 론

요구사항에 내재된 모호성이나 모순성 등을 제거하기 위하여, 복잡한 소프트웨어 시스템의 개발에서는 초기 단계에 정형적인 방법을 이용하여 요구사항을 명세하고 검증하여 왔다. 그러나 요구사항이 복잡해지고, 다양해지면서 정형적인 방법을 이용한 시스템 명세가 방대해지고 이해하기 어려워진다는 문제들이 생겨났다. 따라서 요구사항의 명세를 위해 모듈화 및 객체화 개념 등을 도입하고 있는 추세에 있으며, 특히 복잡한 시스템의 경우에 있어서는 요구사항을 하향식 접근 방법에 의해 분할하고, 이들을 각각 정형적인 방법에 의해 명세하여 합성하는 접근 방법이 이용되고 있다[1, 2].

본 연구에서는 이러한 추세에 따라 요구사항을 계층적으로 모델링하고, 객체지향 개념을 충분히 표현할 수 있는 정형적인 방법의 하나로 객체지향 페트리 넷을 제안하였다. 제안된 객체지향 페트리 넷인 HOONet (Hierarchical Object-Oriented Petri Net)은 모듈화,

† 종신회원 : 한국과학기술원 전자전산학과
jehong@salmosa.kaist.ac.kr
†† 비 회 원 : 한국과학기술원 전자전산학과
icyoon@salmosa.kaist.ac.kr
††† 종신회원 : 한국과학기술원 전자전산학과 교수
bae@salmosa.kaist.ac.kr
논문접수 : 1999년 6월 25일
심사완료 : 1999년 12월 6일

객체화, 추상화, 및 상세화 등의 개념을 지원하도록 개발되었으며, 복잡한 요구사항을 체계적으로 명세할 수 있는 방법을 제공한다. 기존에 객체지향 개념을 지원하는 몇몇 페트리 넷이 요구사항을 명세 및 검증을 위해 제안되었지만 이들은 다음과 같이 복잡한 요구사항의 명세에 유용한 객체지향 개념을 충분히 지원하고 있지 못하고 있다[3-7] .

- 정보 은닉 : 객체 단위별로 데이타와 제어(행위)가 캡슐화되어야 하며, 이들 정보들은 제한된 인터페이스를 통해서만 외부와 상호 작용해야 한다.
- 추상화 : 복잡한 요구사항의 명세를 위해서는 추상화 지원이 필수적이며, 데이타와 제어에 대한, 즉 상태와 행위에 대한 추상화가 모두 가능해야 한다.
- 상속성 : 명세의 융통성 및 효율성을 위해 객체들간에 정보의 상속이 가능해야 하며, 상속성의 제공시, 재정의(overriding) 방법이 함께 제공되어야 한다.
- 메시지 패싱 : 객체간의 상호 작용을 위한 방법으로 메시지 패싱 방법과 이들의 동기적, 비동기적 통신 방법이 제공되어야 한다.
- 다형성 : 객체간의 메소드 호출시 발생할 수 있는 파라메타의 바인딩에 대한 다형성이 제공되어야 한다.

HOONet에서는 이와 같은 개념을 충분히 표현할 수 있는 방법을 제공하며, 특히 요구사항이 부분적으로 제시되었거나, 분석이 전체적으로 완료되지 않은 상황에서도 모델링과 분석이 가능하도록 하였다. 또한 요구사항 명세의 점진적인 합성을 통해 시스템 모델링이 이루어지도록 하였다. 제시한 방법을 이용하여 원자력 발전소의 원자로 안전 제어 시스템의 요구사항[8]을 모델링하고 검증함으로써, HOONet의 적응성을 보였다.

2. 객체지향 페트리 넷

객체지향 페트리 넷의 설계는 페트리 넷의 특성을 객체지향 개념에 접목하는 방법과 객체지향 특성을 페트리 넷 개념에 접목하는 방법으로 크게 구분된다[9]. 이들은 각각 시스템을 모델링하는 서로 다른 특성을 가지고 있지만, 객체지향 개념의 표현은 직관적으로 페트리 넷의 특성을 객체지향 개념 속으로 융합하는 것이 우월하다고 할 수 있다. 이러한 방법에 대한 대표적인 것들은 OPNets[5], LOOPN++[4], COOPN/2[3], 그리고 G-Nets[6] 등이 있다.

2.1 OPNets

OPNets은 실시간 시스템의 모델링을 위한 객체지향 페트리 넷의 일종으로, 객체간의 통신 및 동기화 메카니즘을 객체의 내부 구조와 분리하기 위해 개발되었다[5]. OPNets이 객체들의 계층화된 구조를 통해 추상화 및 상속성을 제공하도록 하고 있으며, "INPORT"와 "OUTPORT"의 제공을 통해 다른 객체들과 상호작용(interaction)을 표현하고자 하고 있으나 객체지향 개념의 지원측면에서 지적할 수 있는 문제점은 (1) OPNets은 데이타와 연산 모두에 대한 진정한 추상화 개념의 지원이라기 보다는 Colored Petri Nets(CPN)에서 제공하는 퓨전(fusion) 메카니즘을 이용하는 모델의 복잡도 감소 방안의 하나라는 점이며, (2) 상속성의 지원에 있어서는 Overriding과 같은 메카니즘의 지원은 전혀 고려하지 못했다는 것이다.

2.2 LOOPN++

LOOPN(Language for Object-Oriented Petri Nets)++에서 객체 구조의 표현은 토큰 클래스와 메소드 클래스에 의해 나타난다. 토큰 클래스에서는 각각의 토큰에 색깔을 부여(coloring) 함으로써, 상속의 근원을 표현하며, 메소드 클래스에서는 퓨전 플레이스(place)와 퓨전 트랜지션을 사용하여 객체간의 상호 작용을 가능하도록 한다. 그러나 LOOPN++가 갖는 단점은 (1) 퓨전의 사용으로 인하여 객체의 정보 은닉에 대한 고려가 미흡하며, (2) 추상화 개념의 지원 측면에서는 토큰 클래스를 통해 데이타에 대한 추상화가 지원되기는 하나 제어(행위)에 대한 추상화는 전혀 제공하지 못한다. (3) 다형성과 비동기적 메시지 패싱에 대해서도 충분히 고려되지 못하였다[4, 10].

2.3 COOPN/2

COOPN/2(Concurrent Object-Oriented Petri Nets/2)는 객체지향 페트리 넷을 이용하여 대규모의 병렬 시스템 명세하고 설계하기 위해 고안된 명세 언어이다. COOPN/2에서의 클래스는 인터페이스를 기술하는 시그너춰(signature) 부분과 객체의 내적인 행위 또는 연산 특성을 나타내는 몸체(body) 부분으로 구성된다. COOPN/2의 중요한 특성은 추상화된 데이타 타입의 계층적인 정의를 통해 매우 다양한 데이타 구조를 토큰으로 가질 수 있도록 하였으며, 객체 접근의 수단이라고 할 수 있는 메소드를 페트리 넷의 인터페이스 트랜지션으로 정의하여 외부와의 상호 작용을 가능하도록 하였다. 객체지향 개념의 반영이라는 측면에서 COOPN/2는 LOOPN++보다 우수하다고 할 수 있으나, COOPN/2 역시 (1) 제어에 대한 추상화 개념은 지원하지 못하고 있

으며, (2) 다형성이나 비동기적 통신 방법에 대해서도 고려하지 못하였다. 특히 (3) 각 객체들의 통합에 대한 시맨틱(semantics)이나 Unfolding 방법이 제공되지 않아 넷의 동작에 대한 해석이 모호하다고 할 수 있다[3].

2.4 G-Nets

G-Nets은 데이타와 제어를 모듈화한 객체지향 페트리 넷으로써, 객체 식별자, 애트리뷰트, 및 메소드의 정의를 나타내는 GSP(General Switching Place)와 메소드의 행위를 페트리 넷으로 묘사하는 IS(Internal Structure)로 구성된다. G-Nets의 특성은 객체가 갖는 캡슐화와 객체간의 메시지 패싱, 객체 인스턴스들의 식별 등을 가능하도록 하며, 메소드의 수행에 대한 종료와 다른 객체의 호출을 위한 메카니즘을 명확하게 제공함으로써, 클라이언트-서버 모델과 같은 복잡한 시스템의 모델에 적용할 수 있도록 하고 있다. 그러나 G-Nets의 개발 의도가 객체간의 상호 작용(통신)을 충분히 지원하기 위한 목적에 있으므로, (1) 데이타 및 제어에 대한 추상화된 표현이 어려우며, (2) 상속성의 지원에 대해서는 전혀 고려하지 못하였다. 또한 다른 객체지향 페트리 넷과 마찬가지로 (3) 다형성이나 비동기적 통신 방법에 대한 표현 방법도 역시 제공하지 못하고 있다[6].

3. HOONet의 정의

3.1 신택스 정의

HOONet은 정보은닉, 추상화, 상속성, 다형성 및 메시지 패싱 등의 객체지향 개념의 지원을 목표로 하는 계층적인 시스템 명세 언어로 개발하였다. HOONet이 갖는 주요한 특성은 제한된 인터페이스를 통한 정보 은닉의 제공과 플레이스, 트랜지션 및 토큰에 대한 추상화 지원이 가능하도록 한 것이다. 이러한 특성들은 복잡하고 대규모의 시스템을 점진적인 방법에 의해 명세하고 분석할 수 있도록 한다[11]. HOONet에 대한 정의는 다음과 같다.

[**정의 1**] 객체지향 페트리 넷, HOONet = (OIP, ION, DD)의 3항목으로 정의한다.

1. OIP(Object Identification Place)는 객체의 유일한 식별자이며, OIP=(*oip, iid, M_0, status*)의 변수들로 정의된다.
   - *oip*는 HOONet으로 모델링된 객체의 이름이며,
   - *iid* (instance identifier)는 객체 인스턴스의 식별자이다. 시스템으로부터 인스턴스가 생성될 때 주어지는 프로세스 식별자, pid와 인스턴스를 생성한 객체의 주소를 나타내는 return으로 구성된다.
   - M_0는 OIP가 갖는 토큰 값을 정의하는 함수이며,
   - *status*는 OIP의 상태를 구별하기 위한 변수로써, "pre" 또는 "post" 값을 갖는다.
2. ION(Internal Object Nets)은 객체의 행위를 표현하는 페트리 넷 모델이다.
3. DD(Data Dictionary)는 객체에 대한 토큰 타입과 변수, 함수 등을 선언하는 부분이다.

DD는 SML 기반의 언어[12]를 이용하여 기술되는데, 각 HOONet 모델별로 하나의 DD가 작성된다.

[**정의 2**] 객체의 행위를 표현하는 ION = (P, T, A, K, N, G, E, F, M_0)로 정의하며, CPN[13]에 기반을 두고 있다.

1. P와 T는 각각 플레이스와 트랜지션의 집합이다.
2. A는 아크(화살표)의 집합으로써, $P \cap T = P \cap A = T \cap A = \varnothing$를 만족한다.
3. K는 임의의 플레이스에 토큰 타입을 정의하는 함수이다.
4. N, G, 와 F는 노드(node), 가드(guard), 아크 표현식(arc expression)에 대한 함수를 의미하며, CPN에서의 정의와 동일하다.
5. F는 임의의 트랜지션에서 OIP로서 아크이며, ION의 테두리로 표시한다.
6. M_0는 임의의 플레이스에 대한 토큰의 초기 마킹(initial marking) 함수이다.

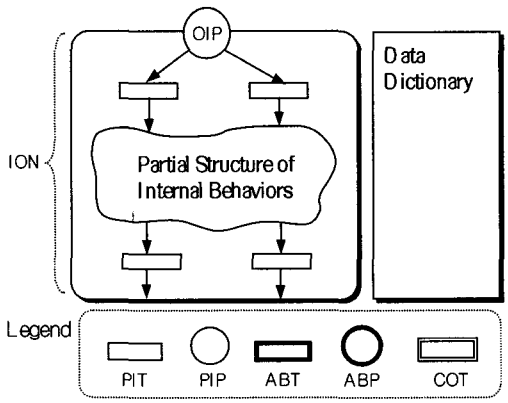

그림 1 HOONet의 기본 구조

그림 1은 HOONet의 구조를 나타낸 것이다. 하나의 객

체는 하나의 HOONet에 의해 표현된다. 하나의 HOONet 모델에서 객체의 인터페이스는 OIP로 국한되며, 내부 정보는 감춰진다. 즉, OIP를 통해 토큰이 유입되고 나간다. 임의의 행위를 수행하기 위해 토큰이 OIP에 마킹되면[*OIP.status* = "pre"], 그 토큰의 값에 따라 Firing될 트랜지션이 선택되어 특정 행위가 수행된다. 수행이 완료되면, 토큰은 "F"를 따라 OIP로 전달된다[*OIP.status* = "post"]. 그림 1에 나타난 Legend는 객체의 내부 행위를 표현하기 위한 기호들로써, 각각 다음의 정의에서 설명한다.

[**정의 3**] HOONet에서의 플레이스의 집합 P = (PIP, ABP)로, 트랜지션의 집합 T = (PIT, ABT, COT)로 구성된다.

1. PIP(PrImitive Place)와 PIT(PrImitive Transition)는 CPN에서 정의하는 일반적인 플레이스와 트랜지션을 각각 의미하며[13],
2. ABP(ABstract Place)는 시스템의 추상화된 상태를 나타내는 플레이스로써, ABP = (*pn, refine_state, action*)으로 정의된다.
   - *pn*은 추상화 플레이스의 이름이며,
   - *refine_state*는 추상화로 표현된 상태가 상세화 되었는지("yes" 또는 "no")를 나타내는 변수이며,
   - *action*은 상세화되지 않은 경우(*refine_state* = "no"), 추상화 상태의 사후 조건(post condition)을 정의한다.
3. ABT(ABstract Transition)은 시스템의 추상화된 동작을 나타내며, ABT = (*tn, refine_state, action*)으로 정의된다. 이들 항목의 의미는 ABP의 정의에서와 동일한 개념을 갖는다.
4. COT(COmmunicative Transition)는 객체간의 통신을 표현하는 트랜지션으로써, COT = (*tn, target, c_type, action*)으로 정의된다.
   - *tn*은 트랜지션의 이름이며,
   - *target*는 COT에 의해 호출되는 객체가 모델링 되었는지를("yes" 또는 "no") 나타내는 변수이며,
   - *c_type*은 객체간의 통신이 동기적("SYNC")인가, 비동기적("ASYN")인가를 나타내며,
   - *action*은 *target* = "no"인 경우, 호출할 메소드의 사후 조건을 정의한다.

HOONet의 DD에 정의되는 토큰의 타입은 크게 기본형(primitive type)과 복합형(compound type)으로 구분된다. 기본형은 boolean이나 integer 등과 같은 기본 타입을 의미하며, 복합형은 기본형의 조합으로 재 정의될 수 있다. 복합형은 일반적으로 추상화된 토큰 타입을 정의하기 위해 사용되는데, 추상화 상태에서는 "Complex" 타입으로 선언되며, 상세화 상태에서는 기본형의 조합으로 재정의하여 선언하게 된다. 추상화 상태에서 토큰의 정보를 상세하게 정의할 수 도 있으나, 이는 행위의 상세화 과정에서 변경되기 쉽기 때문에 피하는 것이 좋다.

3.2 시맨틱 정의

HOONet이 갖는 시맨틱을 정의하기에 앞서 몇가지 용어를 간략히 정의한다.

- $Var(t)$는 모든 트랜지션 t에서의 변수들의 집합이며, $Var(expr)$은 표현식 $expr$에 나타나는 모든 변수들의 집합을 의미한다.
- $E(x_1, x_2)$는 (x_1, x_2)에 대한 표현식이다. 즉, $\forall (x_1, x_2) \in (P \times T \cup T \times P)$에 대하여 $E(x_1, x_2) = \sum_{a \in A(x_1, x_2)} E(a)$. 여기서 A는 아크의 집합을 의미한다.
- $E(p,t)<b>$는 p에서 t 상에 나타나는 표현식이 바인딩 b에 의해 평가(evaluation)됨을 의미한다.
- $K(p)$는 플레이스 p에 토큰 타입의 집합을 할당하는 함수이고, $M(p)$는 선언된 토큰 타입에 따라 플레이스 p에 토큰 값을 지정하는 함수이다.
- x가 임의의 플레이스 또는 트랜지션이라 할 때, $\bullet x$는 x로의 입력을 의미하며, $x \bullet$는 x로의 출력을 의미한다. 즉, $\bullet x = \{y \in P \cup T \mid (y, x) \in A\}$ 이고 $x \bullet = \{y \in P \cup T \mid (x, y) \in A\}$이다.

일반적으로 페트리 넷의 설계에서 시맨틱의 정의는 바인딩(binding), 마킹(marking), 트랜지션의 Enabling과 Firing에 의해 정의된다. HOONet에서의 기본적인 시맨틱은 K. Jensen의 CPN 시맨틱[13]을 따른다. 그러나 HOONet에서의 행위적 시맨틱(behavioral semantics)은 이러한 기본적인 시맨틱을 이용해 재정의해야 할 필요성이 있다. 이들은 HOONet 구성 요소들의 Enabling과 Firing 시맨틱을 정의하는 것이다.

[**정의 4**] OIP의 행위적 시맨틱 BS(OIP)는 다음과 같이 정의한다.

$$BS(OIP) = \begin{cases} \forall t \in OIP \bullet,\ E(OIP, t)<b> & \text{iff } status = "pre" \\ M(x \bullet) \text{ for } x = OIP.iid.return & \text{iff } status = "post" \end{cases}$$

[주] 만약 status의 값이 "pre"인 경우에는 OIP에서

의 출력 트랜지션들에 대하여 Firing할 트랜지션을 선택하게 되고, "post"인 경우는 자신을 호출했던 객체로 토큰을 반환한다. 그러나 만약 return의 값이 "self"인 경우는 자신이 특정 행위를 시작하였기 때문에 행위를 종료한다.

[**정의** 5] ABP와 ABT의 행위적 시맨틱 BS(ABP)와 BS(ABT)는 각각 다음과 같이 정의한다.

$$BS(ABP)=\begin{cases} M(OIP) \text{ for } ABP.pn = OIP.oip \\ \qquad\qquad \text{iff } refine\_state=\text{"true"} \\ R(action);\ \forall t \in ABP\bullet,\ E(ABP, t)<b> \\ \qquad\qquad \text{iff } refine\_state = \text{"false"} \end{cases}$$

[주] 추상화 상태(플레이스)가 상세화 되었다면, 상세화된 모델로 토큰이 전달되며, 그렇지 않은 경우는 *action*에 정의된 조건식을 수행하고(R(action)), ABP로부터의 출력 트랜지션들에 대하여 바인딩을 수행한다.

$$BS(ABT)=\begin{cases} M(OIP) \text{ for } ABT.tn = OIP.oip \quad \text{iff } refine\_state=\text{"true"} \\ R(action);\ +M(ABT\bullet)\wedge\text{-}M(\bullet ABT) \text{ iff } refine\_state = \\ \qquad\qquad \text{"false"} \end{cases}$$

[주] 여기서, $+M(ABT\bullet)\wedge\text{-}M(\bullet ABT)$는 트랜지션 ABT의 Firing에 의해 ABT의 출력 트랜지션에 토큰을 추가하고, 입력 트랜지션으로부터 토큰을 제거함을 의미한다.

[**정의 6**] COT에 대한 행위적 시맨틱 BS(COT)는 다음과 같이 정의된다. 객체간의 상호작용이 동기적[c_type = "SYNC"]으로 이루어지는 경우

$$BS(COT)=\begin{cases} M(OIP) \text{ for } COT.tn = OIP.oip;\ \text{-}M(\bullet COT) \\ \qquad\qquad \text{iff } target = \text{"true"} \\ R(action);\ \ +M(COT\bullet)\wedge\text{-}M(\bullet COT) \ \text{ iff } target = \\ \qquad\qquad \text{"false"} \end{cases}$$

또한 비동기적인 상호 작용[*c_type* = "ASYN"]의 경우에는 다음과 같이 정의된다.

$$BS(COT) = \begin{cases} M(OIP) \text{ for } COT.tn = OIP.oip;\ +M(COT\bullet)\wedge \\ \qquad\qquad \text{-}M(\bullet COT) \text{ iff } target = \text{"true"} \\ +M(COT\bullet)\wedge\text{-}M(\bullet COT) \quad \text{iff } target = \text{"false"} \end{cases}$$

[주] 속성 변수 target = "true"인 비동기적인 통신의 경우에는 토큰이 복사되어 하나는 호출되는 객체로 전달되며, 다른 하나는 출력 플레이스로 전달된다.

이와 같이 정의된 HOONet의 시맨틱에 기반하여 시스템의 기능적 행위를 명세할 수 있다. 이상의 정의들을 이용하여 시스템을 명세하는 과정은 다음 절에서 예제 시스템을 통해 설명하기로 한다.

4. 요구사항 분석 및 명세

HOONet을 이용하여 시스템을 명세하기 위한 절차가 그림 2와 같다. 먼저, 자연어 기반의 요구사항을 행위의 주체인 액터(actor) 중심으로 분해한다. 분해된 요구사항은 테이블 구조를 갖는 스크립트(script)에 의해 표현되며, 이는 HOONet으로 번역되어 명세된다. 명세한 HOONet의 정확성을 검증하기 위해 모델을 분석한다.

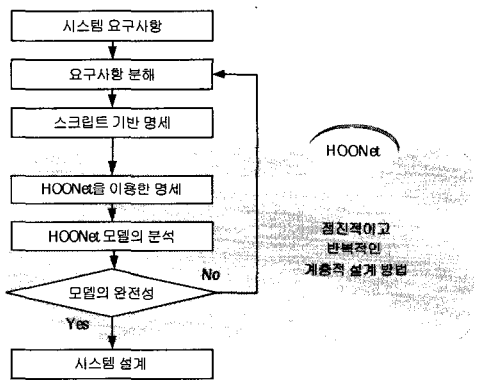

그림 2 HOONet을 이용한 요구사항 명세 절차

4.1 시스템 요구사항

HOONet을 이용한 시스템 명세 절차를 설명하기 위해 원자력 발전소의 원자로 정지 시스템(shutdown system)의 소프트웨어 부분에 대한 요구사항을 이용하였다[8]. 이에 대한 요구사항을 살펴보면 다음과 같이 표현할 수 있다. 기술된 요구사항은 이해의 편의상 일부의 요구사항을 단순화 한 것이다.

「원자로 안전 시스템(reactor safety system)의 Shutdown Trip System(STS)은 온도와 압력이라는 입력 값을 받아들여 Turbine, Emergency Core Cooling(ECC) system, Heat Transport Pump(HTP)를 제어한다. 온도와 압력이 정상 범위일 때, STS는 동작하지 않으며, 정상의 범위를 벗어나면 Turbine, ECC, HTP의 동작이 변화시키는 다운(down) 절차에 진입한다. 온도와 압력이 정상의 범위에 돌아오면, Turbine, ECC, HTP가 정상적 동작 모드로 복귀한다.」

4.2 요구사항 분해 및 스크립트 명세

요구사항의 1차적인 명세을 위해 스크립트를 이용한

표 1 STS 시스템의 스크립트 명세

<Level 1 : Trip>					
Action Name	*Actors*	*Pre_condition*	*Stimulus*	*Response*	*Post_condition*
정상동작	STS	Turbine, ECC, HTP: 정상 동작	온도 압력: 정상 범위	No action	Turbine, ECC, HTP: 정상 동작
비정상동작	STS	Turbine, ECC, HTP: 정상 동작	온도 XOR 압력: 비정상	Down	Turbine, ECC, HTP: 동작 변화
복구동작	STS	Turbine, ECC, HTP: 동작 변화	온도 압력: 정상 범위	Recover	Turbine, ECC, HTP: 정상 동작

다. 스크립트는 요구사항을 분해하고, 이들로부터 얻은 정보를 테이블 형태로 표현한 것으로 다음과 같은 정의하였다.

<Script Level : Behavior Name>
{ Action Name;
Coordinating and Participating Actors;
Pre_condition;
Stimulus;
Response;
Post_Condition
}

앞서 기술한 원자로 안전 시스템의 STS의 요구사항 분해를 통해 얻은 정보를 스크립트를 이용하여 명세하면 표 1과 같다.

표 1의 스크립트 Level 1의 명세는 STS의 최상위 수준에 대한 명세이다. 따라서 Turbine, ECC, HTP 등에 대한 기능 명세와 Down 및 Recover에 대한 세부적인 기능 명세가 스크립트의 하위 수준에서 명세될 수 있다.

4.3 HOONet을 이용한 명세

스크립트에 의해 명세된 요구사항은 다음과 같은 매핑 규칙에 의거하여 HOONet으로 표현할 수 있다.

스크립트 요소	HOONet 요소
Script level	Level of hierarchy
Behavior name 또는 Actor	Name of object
Action name	Name of methods within an object
Pre_condition	Token value of input places
Stimulus	Transition
Response	Expressions of out arcs
Post_condition	Token value of output places

이와 같은 규칙에 따라 Level 1: Trip을 HOONet으로 표현한 모델이 그림 3에 있다. 그림 3에서 보는바와 같이 온도와 압력의 상태에 따라 normal, recovered, abnomal의 상태로 전이될 수 있다. "recovered" 상태는 원자로가 정상 상태로 복귀하였을 때, ECC, HTP, Turbine의 상태가 정상화 되었음을 나타내기 위한 것이다. 비정상(abnormal) 상태에서의 다운 절차를 추상화된 트랜지션으로 표현하였다. 이러한 추상화된 표현은 모델을 간단하게 명세할 수 있도록 하며, 복잡한 요구사항을 갖는 시스템을 계층적으로 모델링을 할 수 있도록 한다.

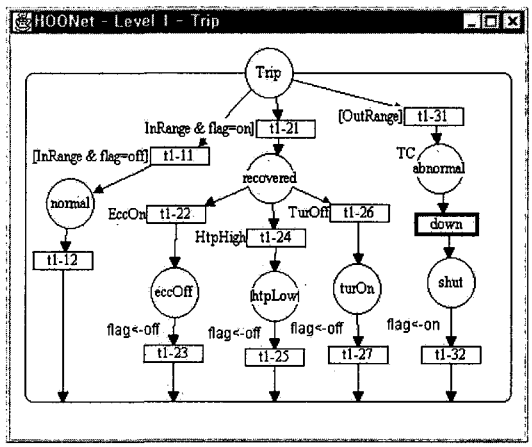

그림 3 Level 1: Trip의 HOONet 표현

추상화 트랜지션 "down"은 명세 수준 Level 2에서 하나의 상세화된 HOONet에 의해 표현되는데, 이에 대한 모델이 그림 4에 나타나 있다. 원자로의 이상 현상을 방지하기 위하여 ECC, HTP, Turbine의 동작 상태를 제어하는 부분이다.

이와 같이 추상화된 컴포넌트가 상세화되기 위해서는 이들간에 모피즘(morphism)이 성립해야 한다. HOONet

에서의 모피즘의 만족은 추상화된 컴포넌트와 상세화된 모델간의 행위에 대한 동등함(behavioral equivalence)을 보여주는 것으로써, 조건 1을 만족시키는 것으로 충분하다.

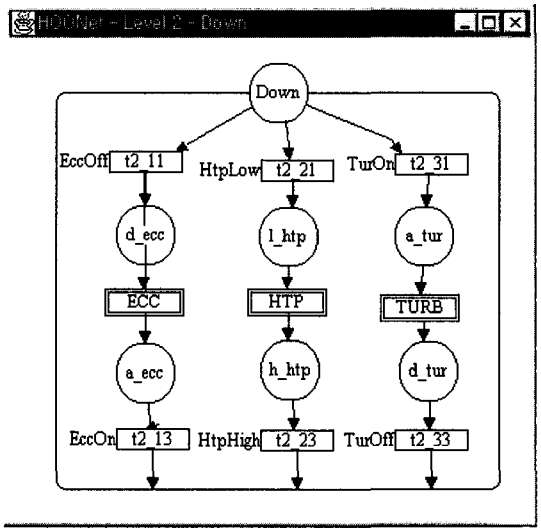

그림 4 Level 2: 추상화 트랜지션 "down"의 상세화 모델

[**조건 1**] 추상화 컴포넌트 ABP, ABT가 상세화될 때, 이들간의 모피즘을 위해서는 다음과 같은 조건이 만족되어야 한다.

1. 추상화 플레이스 ABP에 대하여
   - 토큰 타입 : $\forall p \in ABP,\ K(p) \subseteq TT_{RN}$
   - 입력 아크의 표현식(a_i)과 출력 아크의 표현식(a_o) : $\bigcup_{a_i, a_o} Var(\exp r) \subseteq Var(OIP_{RN}) \subseteq V_{RN}$
2. 추상화 트랜지션 ABT에 대하여
   - 토큰 타입 : $\forall t \in ABT,\ \{K(\bullet t) \cup K(t \bullet)\} \subseteq TT_{RN}$
   - a_i와 ABT $\bullet$의 a_o : $\bigcup_{a_i, a_o} Var(expr) \subseteq Var(OIP_{RN}) \subseteq V_{RN}$
   - 가드 함수(guard function) : $G(t) \equiv \bigcup_{\forall t_i \in (OIP_{RN})_\bullet} G(t_i)$

조건 1에서 TT_{RN}과 V_{RN}은 각각 상세화된 모델(RN)의 DD에서 정의된 모든 토큰 타입과 모든 변수의 집합을 의미한다.

다형성 추상화된 컴포넌트가 상세화되었을 때[refine_state = "true"], 넷의 행위는 [정의 5]에서 설명한 것과 같이 토큰을 상세화된 모델로 전달하는 것이다. 이때, 추상화된 토큰은 상세화된 토큰으로 분해되어야 하는데, 이 과정에서 파라메타에 대한 다형성(parametric polymorphism)이 발생한다. 예를 들어 그림 3의 플레이스 "abnormal"의 토큰 타입 "TC"는 STS의 트립(trip) 조건을 나타낸다. 즉, TT TC = complex with weak | strong;과 같이 정의된다. 토큰이 상세화된 모델 - 그림 4의 "Down" - 로 전달될 때, "TC"의 토큰 값이 온도(Temp)에 대하여 "weak" 이면, 토큰 "TC"는

```
TT TC = record with {
            Temp = OutRange | InRange;
            Pres  = OutRange | InRange;
        }
```

로 분해되며, {(Temp,OutRange), (Pres, InRange)}의 값이 배정되고, 압력(Pres)에 대하여 "weak"이면, {(Temp, InRange), (Pres, OutRange)} 값이 상세화된 토큰의 값으로 주어진다.

상호 작용을 위한 메시지 패싱 그림 4의 "Down" 객체는 원자로의 다운(shutdown)을 위한 절차를 진행한다. 이 절차는 Turbine을 정지시키고, ECC 시스템을 가동시키고, 또는 HTP의 속도를 높이는 동작으로 구성된다. 원자로의 온도를 낮추기 위해 ECC 시스템을 가동시키는 액션에 대하여 그림 5에 모델링 하였다.

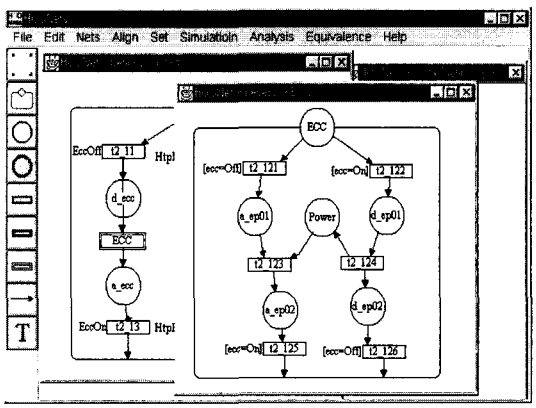

그림 5 Level 2: Down 객체와 ECC 객체간의 상호 작용 모델링

그림 5에서 COT "ECC"는 객체 ECC의 메소드를 호출하기 위한 트랜지션이다. 이 트랜지션에 의해 메소드가 호출되면, 특정 값을 지닌 토큰은 ECC 객체로 패싱된다. ECC 객체에서 메소드의 수행이 완료되면 토큰은 Down 객체의 호출한 지점으로 반환된다.

상속을 통한 객체의 상세화 상속성은 객체들의 계층

관계를 기반으로 속성(attributes)과 연산(operations)을 공유하기 위한 메카니즘이다[7]. 따라서 상위 객체(superclass)에서 정의된 특성(properties)을 하위 객체(subclass)에서 모두 상속하여 사용할 수 있다. 그림 6은 그림 4의 "Down" 객체에서 정의된 모든 특성을 상속받은 Level 3의 "Down2" 객체를 모델링한 것이다. 이 객체에서는 상속받은 특성 이외에 자신이 고유하게 정의하는 행위 - 안전 시스템의 다운 절차 과정에 알람(alarm) 기능을 수행하는 동작 - 를 추가로 정의하고 있다. 상위 객체로부터 상속받은 특성은 "Down" 플레이스에 의해 표현되었는데, 이 부분은 - 그림 6에서의 t3_11 > Down > t3_12 부분 - 이해의 편의를 위한 표현(syntactic sugar)이기 때문에 생략할 수 있다.

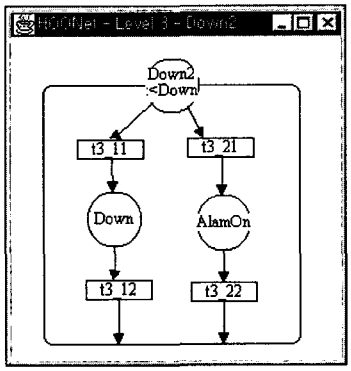

그림 6 Level 3: 객체 "Down"의 특성을 상속받아 정의된 객체 "Down2"

상속을 이용하여 하위 객체의 행위를 상세화할 때, Overriding 메카니즘이 고려될 수 있다. 이는 상속받은 특성의 일부를 수정하여 하위 객체의 행위를 변경하기 위한 방법이다. Overriding 메카니즘을 제공하기 위해서는 Overridden 메소드와 Overriding 메소드 중, 어느 메소드를 바인딩할 것인지를 결정해야 한다. HOONet에서는 이러한 메카니즘을 제공하기 위하여 다음의 조건 2를 만족하도록 하였다.

[조건 2] 상속성의 제공을 위해 상위객체(S)와 하위객체(I), 그리고 하위객체에 새롭게 (재)정의되는 메소드(N)간에 다음과 같은 조건이 만족되어야 한다.

1. 토큰 타입의 집합 : $TT_I = TT_S \cup TT_N$
2. Enabled 트랜지션의 집합

$$ET_I = \{t_i \mid \forall t_{N,E}(OIP_I,t)\langle b\rangle\} \cup \{t_j \mid \forall t_{S,E}(OIP_S,t)\langle b\rangle\}$$

3. $t_i \in T_N,\ t_j \in T_S$에 대하여,

$$\exists t_i, t_j \in ET_I \mapsto ((t_i, b) \in Y \wedge (t_i, b) \notin Y)$$

위의 정의에서 Y는 페트리 넷의 스텝 시맨틱을 의미하는 것으로 Enabled 트랜지션으로부터 Firing하기 위한 트랜지션을 결정한다[13].

이상에서와 같이, 원자로 정지 시스템의 단순화된 요구사항을 HOONet을 이용하여 명세하고, 특히 HOONet이 객체지향 개념을 어떻게 표현하는 가에 주안점을 두고 설명하였다.

4.4 모델의 분석을 위한 Unfolding

명세된 HOONet 모델에 대한 교착상태(deadlock), 생존성(liveness), 공정성(fairness) 등을 분석하기 위해서는 새로운 분석 방법을 제시하거나 또는 기존에 증명된 일반화된 페트리 넷으로 Unfolding 되어져야 한다. 본 연구에서는 후자의 방법을 채택하여 HOONet 모델을 CPN 모델로 Unfolding하는 방법을 제시하였다. Unfolding은 HOONet의 여러 구성요소를 CPN과 동등한 표현으로 펼침으로써, CPN에서 제공하는 분석 방법을 그대로 이용하기 위함이다. HOONet 모델에서 Unfolding의 대상은 OIP, ABP, ABT 그리고 COT 등이다. 본 논문에서는 지면상 OIP와 ABT에 대한 Unfolding 방법에 대하여 설명한다. ABP나 COT에 대한 Unfolding은 OIP와 ABT의 Unfolding으로부터 유도될 수 있으며, [11]에 자세히 기술되어 있다.

OIP Unfolding HOONet의 Unfolding은 기본적으로 OIP Unfolding 과정을 선행적으로 거쳐야 한다. OIP Unfolding은 그림 7에 나타난 것처럼 프레임 "F"를 제거하고, OIP를 복사하여 넷의 출구로 사용하는 것이다.

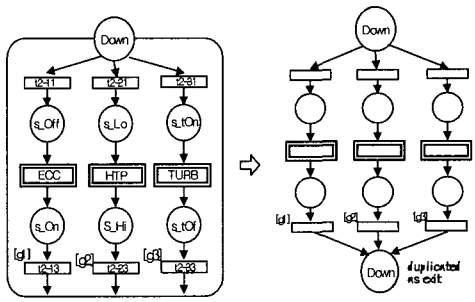

그림 7 OIP Unfolding 과정

ABT Unfolding 추상화된 컴포넌트 ABT의 Unfolding은 2단계에 걸쳐 이루어지는데, (1) 추상화 트

랜지션을 두 개의 서브 트랜지션으로 분리하고, (2) 두 개의 서브 트랜지션 각각을 Unfolding된 상세화 넷(추상화 컴포넌트의 상세 모델)의 시작 플레이스와 종료 플레이스에 연결시킨다. 그림 8은 그림 3에 나타난 추상화 트랜지션 "down"의 Unfolding을 보여준다. 이 그림에서 분리된 두 개의 트랜지션 "i_down"과 "o_down"은 각각 상세화된 넷의 Invocation과 이의 종료를 의미한다.

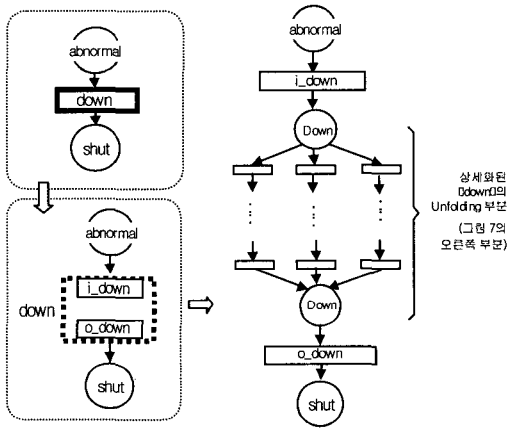

그림 8 ABT Unfolding 과정

이와 같은 HOONet의 Unfolding은 시스템의 광역 구조(global structure)를 표현하기 위해 사용될 수 있으며, Unfolding되어진 HOONet 모델은 기존의 도달성 분석(reachability analysis) 방법[13]을 이용하여 요구사항에서의 교착상태, 생존성 등을 검사할 수 있다[5.3절 참조].

5. 합성적 모델 분석 및 검증

도달성 분석은 페트리 넷 모델의 모든 도달 가능한 상태들을 방향성 그래프(Directed Graph)에 의해 표현하고, 이를 이용하여 요구사항에 대한 특성을 분석한다. 도달성 그래프를 이용하여 페트리 넷을 분석하는 것은 시스템에 대한 행위 분석을 수행하기 위한 것으로 시스템 모델에 도달 가능하지 않는 상태가 존재하는가, 특정한 행위의 실행으로 인하여 다른 행위의 실행이 불가능하게 되는 경우가 있는가, 그리고 비결정성에 의한 실행 경로의 결정이 불가능해 지는 경우가 있는가 등을 검사하기 위한 것이다. 본 연구에서 제시하는 도달성 분석의 특징은 전체 요구사항에 대한 HOONet 모델이 모두 생성되지 않았다고 하더라도 도달성 분석이 가능하다는 것이다. 이는 각 HOONet이 OIP를 인터페이스로 하는 모듈화된 페트리 넷이며, "ONE-ENTRY ONE-EXIT" 조건을 만족하고 있기 때문이다[14]. 따라서 복잡하고 대규모의 시스템에 대한 요구사항이 부분적으로 기술되었거나, 또는 전체적인 분석이 이루어지지 않은 상황에서도 HOONet을 이용하여 시스템의 계층적인 모델링이 가능하며, 점진적인 도달성 분석이 가능하다.

5.1 도달성 그래프의 생성

HOONet 모델에 대한 도달성 그래프의 생성 방법에 알고리즘 1에 제시하였다. 이 알고리즘에서는 HOONet의 부분적인 명세만을 이용하여 도달성 분석이 가능하도록 하고 있다. 다시 말해서 모델내에 상세화 되지 않은 추상화 컴포넌트가 포함되어 있다고 하더라도 도달성 그래프를 생성할 수 있도록 하였다. 알고리즘의 간략한 표현과 이해를 돕기 위해 다음의 두 함수를 정의하였다.

- GenNode(M_1,M_2) : 이 함수는 하나의 노드 M_2를 생성하고, 이를 도달성 그래프에 존재하는 노드 M_1과 연결시킨다. 그리고 M_1 노드는 "R"로 표시하고, M_2는 "M"으로 표시한다. 그래프에서 "R"은 도달 가능한 노드를 의미하며, "M"은 아직 후속자(successor)를 갖지 않는 노드임을 의미한다.
- AllGen(M_1, M_i) : 노드 M_1으로부터 모든 후속자 M_i를 찾고, 이들 M_i를 노드 M_1과 연결시킨다. 그리고 M_1은 "R"로 표시하고, 모든 M_i는 "M"으로 표시한다.

[알고리즘 1] GenRGraph(input: set of HOONet models output: RG)

1. Let the node which has initial marking, be the root node of a RG and then tags this node with "M";
2. While all nodes tagged with "M"
   2.1 OIP: if (*status* = "pre") then AllGen(M_1,M_i);
   else if (*return* = "self") then Exit();
   else GenNode(M_1, M_2) for node pointed by return;
   2.2 PIP: AllGen(M_1, M_i);
   2.3 ABP or ABT: if (*refine_state* = "false") then AllGen(M_1, M_i);
   else GenNode(M_1, M_2) for OIP of the refined net;
   2.4 COT: if (*target* = "no") then AllGen(M_1, M_i);
   else if (*c_type* = "SYNC") then GenNode(M_1, M_2) for OIP of the called net;
   else AllGen(M_1, M_i); || GenNode(M_1, M_2) for OIP of the called net;
3. Until (No more "M" exist);
4. End of Algorithm

5.2 도달성 그래프의 합성

HOONet 모델 각각에 대한 도달성 그래프가 생성되었다면, 이들의 합성을 통해 광역 도달성 그래프를 생성할 수 있다. 도달성 그래프의 합성을 통해 모델을 분석하는 방법에 대하여 기존에도 몇가지 연구가 제시되었다[2, 15]. 이러한 기존의 연구는 시스템을 구성하는 서브 시스템별로 분석이 이루어지기 때문에 전체 시스템에 대한 분석이라고 볼 수 없다. 즉, 모든 서브 시스템에 대한 모델링이 이루어진 뒤에 전체 시스템에 대한 분석이 가능하다. 그러나 본 논문에서 제시하는 합성적 분석 방법은 전체 시스템을 계층적으로 분할하여 모델링하기 때문에 추상화된 상태에서의 전체 시스템에 대한 분석이 가능하다. 즉, 하향식의 계층적 분할에 의거한 모델링은 서브 시스템간의 상호 작용을 고려한 모델링이 가능하고, 누락되었거나 분석되지 않은 부분에 대해서는 추상화로 표현하기 때문에 전체 시스템에 대한 시뮬레이션이 가능하고, 점진적인 모델링을 통해 도달성 그래프의 합성이 이루어질 수 있다. 이는 전체 시스템에 대한 분석과 검증이 모델이 완성되지 않은 상태에서도 가능함을 의미한다.

개별적으로 생성된 HOONet 모델의 도달성 그래프를 합성하는 방법이 알고리즘 2에 제시되었다. 이 알고리즘에서 "*SetABC*"는 모든 ABPs, ABTs, 그리고 COTs의 집합이고, "M_i"는 도달성 그래프에서의 임의의 노드 "abc_i"의 선행자(predecessor)이다.

[알고리즘 2] MergeGraphs(input: RGs, output: GRG)

1. Select a component $abc_i \in SetABC$
2. Using the algorithm 1, generate the reachability graph *corresponding to the refined model of the* abc_i
3. Inserts the reachability graph at the next location of the M_i

그림 9는 Level 1의 Trip 객체와 Level 2의 Down 객체에 대한 도달성 그래프에 대한 합성 과정을 알고리즘 2에 의거하여 개념적으로 보여준 것이다. 이러한 합성 과정의 반복을 통해 전체 시스템에 대한 도달성 그래프를 완성할 수 있다.

5.3 모델의 검증

도달성 그래프에 의한 HOONet 모델의 검증은 시스템 모델의 행위적 특성이 요구사항에 따라 정확히 모델링되었는가, 그리고 요구사항에 존재하는 모순성은 없는가 등을 검사하기 위한 것으로 다음과 같은 특성들을 대표적으로 검증할 수 있다.

교착상태의 검사 도달성 그래프에서 교착상태는 Enabled

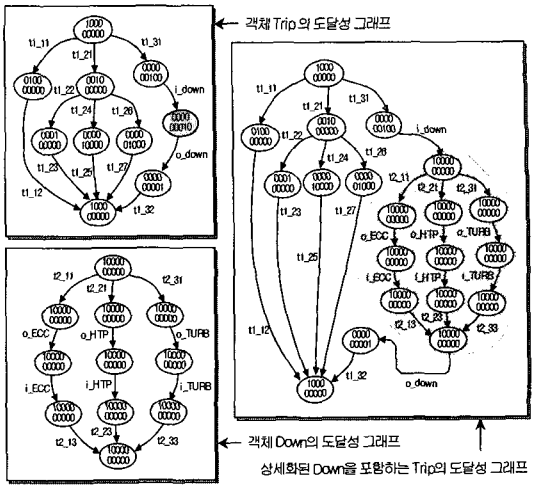

그림 9 합성에 의해 광역 도달성 그래프의 생성

트랜지션이 존재하지 않는 데드 마킹(dead marking)에 의해 검사될 수 있다. 초기 마킹에서 데드 마킹으로의 전이와 더 이상의 전이가 발생하지 않는 상태는 시스템의 행위가 교착 상태에 있음을 의미한다.

생존성 검사 모든 트랜지션은 적어도 한번 Firing 되어야 하기 때문에 HOONet 모델에 존재하는 모든 트랜지션은 도달성 그래프에 나타나야 한다. 그래프에 나타나지 않는 트랜지션이 존재한다는 것은 시스템이 그 상태에 도달할 수 없음을 의미한다.

공정성 검사 임의의 트랜지션의 Firing 횟수가 유한하다면 이는 시스템에 무한 루프(infinite loop)가 존재하지 않음을 의미한다. 따라서 도달성 그래프에서도 무한히 반복되는 부분 그래프(partial graph)가 존재하지 않는다.

비 결정성 검사 하나의 입력 플레이스에 대하여 두 개의 트랜지션이 Enable 되어질 때, 비결정적인 선택이 이루어져야 한다. 이러한 문제는 요구사항의 부정확성으로부터 기인된다고 볼 수 있으며, 도메인 전문가에 의해 해결되어야 한다.

6. 결 론

소프트웨어 시스템의 요구사항을 정형적인 방법으로 명세하는 것은 모델의 검증을 통해 요구사항의 결함을 검출하기 위한 것이다. 요구사항을 기술하기 위한 정형적인 방법중에서 객체지향 페트리 넷은 객체지향 파라다임의 유용한 개념을 이용하여 복잡하고 대규모의 시스템을 직관적이고 실제적으로 모델링할 수 있도록 한

다.

본 논문에서는 객체지향의 유용한 개념들을 충분히 지원할 수 있는 계층 구조의 객체지향 페트리 넷인 HOONet을 제안하고, 이를 이용하여 원자로 안전 시스템의 Shutdown Trip System을 모델링 하였다. HOONet 모델링의 장점은 복잡한 요구사항의 하향식 분할을 기반으로 시스템의 계층적 모델링이 가능하며, 특히 요구사항이 부분적으로 제시되었거나, 전체 요구사항이 분석되지 않았다고 하더라도 시스템의 모델링과 도달성 분석이 이루어질 수 있도록 한 것이다. 이는 전체 시스템에서 관심있는 부분에 대해서만 모델링과 분석을 수행할 수 있음을 의미한다. 현재, HOONet 기반의 모델링 및 분석을 지원하기 위한 도구가 *Visual Cafe*^{TM}을 이용하여 개발 중에 있다. 제시한 HOONet 기반의 시스템 명세 및 검증 방법은 다양한 분야에서 적용되고 있는 시나리오 기반의 시스템 모델링에 쉽게 적용할 수 있으며, 페트리 넷의 시뮬레이션을 이용하여 *On-the-fly* 방식에 의한 변경 분석 등을 수행할 수 있을 것이다. 또한 HOONet의 적용성 및 실용성을 향상시키기 위해서는 실시간 제약사항을 모델링하기 위한 Timed HOONet으로의 확장이 연구되어야 한다.

참 고 문 헌

[1] E. Battiston, et al, "An Incremental Specification of a Hydroelectric Power Plant Control Systems using a Class of Modular Algebraic Nets," *Proc. of the 16th Int'l Conf. on ATPN'95, Also in LNCS 935*, pp.84-102

[2] S. Cheung and J. Kramer, "Context Constraints for Compositional Reachability Analysis, *ACM TOSEM*, Vol.5, No.4, Oct., 1996, pp.334-377

[3] O. Biberstein, D. Buchs, and N. Guelfi, "Modeling of Cooperative Editors Using COOPN/2," *Proc. of Int'l Workshop on OOP&MC*, Osaka Japan, June, 1996

[4] C. Lakos and C. Keen, "LOOPN++: A New Language for Object-Oriented Petri NEts," *Technical Report R94-4,* Networking Research Group, University of Tasmania, Australia, April, 1994

[5] Y.K. Lee and S.J. Park, "OPNets: An Object-Oriented High-Level Petri Nets for Real-Time System Modeling," *Journal of Systems and Software*, Vol(20), 1993, pp.69-89

[6] A. Perkusich and J.C.A Figueiredo, "G-Nets: A Petri Net Based Approach for Logical and Timing Analysis of Complex Software Systems," *Journal of Systems and Software,* Vol(39), 1997, pp.39-59

[7] J. Rumbaugh, et al, *Object-Oriented Modeling and Design*, Prentice Hall, 1991

[8] AECL CANDU, *Software Requirement Specification - Wolsong NPP 2,3,4,* 86-68350- SRS-001, Canada, June 1993

[9] R. Bastide, "Approaches in Unifying Petri Nets and the Object-Oriented Approach," *Proc. of the Int'l Workshop on OOP&MC*, Turin Italy, June, 1995, *http://wrcm.dsi.unimi.it/PetriLab/ws95/home.html*

[10] C. Lakos, "The Object Orientation of Object Petri Nets," *Proc. of Int'l Workshop on OOP&MC*, Turin Italy, June, 1995

[11] J.E. Hong and D.H. Bae, HOONets: Hierarchical Object-Oriented Petri Nets for System Modeling and Analysis, *CR-TR-98-132*, Dept. of Computer Science, KAIST, Nov., 1998, *http://cs.kaist.ac.kr/library/tr/*

[12] J.D. Ullman, *Elements of ML Programming*, Prentice Hall, 1998

[13] K. Jensen, *Colored Petri Nets Vol. I and Vol. II*, Springer-Verlag, 1992

[14] I. Suzuki and T. Murata, "A Method for Stepwise Refinement and Abstraction of Petri Nets," *Journal of Computer and System Sciences*, Vol.27, 1983, pp.51-76

[15] W.J. Yeh and M. Young, "Compositional Reachability Analysis Using Process Algebra," *Proc. of the Int'l Symposium on TAV*, 1991, pp.49-59

홍 장 의

1988년 충북대학교 전산통계학과 학사. 1990년 중앙대학교 전자계산학과 석사. 1990년 ~ 1998년 한국국방연구원(국방정보체계연구소) 선임연구원. 1999년 ~ 현재 국방과학연구소 선임연구원. 1995년 ~ 현재 한국과학기술원 전자전산학과 전산학전공 박사과정. 관심분야는 소프트웨어공학, 객체지향 모델링, 정형화 기법

윤 일 철

1999년 서강대학교 전자계산학과 학사. 1999년 ~ 현재 한국과학기술원 전자전산학과 전산학전공 석사과정. 관심분야는 소프트웨어공학, 컴포넌트 기반의 소프트웨어 개발 방법론, 분산 트랜잭션 처리, 정형화 기법

배 두 환

정보과학회논문지 : 소프트웨어 및 응용
제 27 권 제 1 호 참조