공학 연구논문집 제27권 제2호 pp. 21~32, 울산대학교
Journal of Engineering Rrearch Vol. 27, No. 2, pp. 21~32, 1996
University of Ulsan.

실시간 시스템의 설계 지원을 위한 CODARTS에서 Petri Net으로의 변환에 대한 연구

정민포 · 김규년
전자계산학과

<요 약>

본 논문에서는 CODARTS의 산물중 Task Architecture Diagram에서 Petri Net으로의 변환 방법을 제안한다. CODARTS는 여러 개의 동시 작업(Concurrent Task)으로 이루어진 시스템을 설계할 때 사용되는 방법이다. 하지만, CODARTS에서 생성되는 산물(product, output)은 정적인 측면만을 강조하게 된다. 그래서, 시스템을 설계한 후 성능 분석을 할 때, 시스템 요소들 사이의 동적인 측면을 강조하기 위해 Petri Net으로 변환을 함으로써 더 나은 실시간 소프트웨어의 설계 및 분석을 유도할 수 있다.

The Study on the transformation from CODARTS to Petri Net for the design support of Real-Time System

Min-Po Jung · Kyoo-Nyun Kim
Dept. of Computer Science

<Abstract>

In this paper, we propose the transformation method from the TAD of CODARTS to Petri Net. CODARTS is used to design a system consisting of concurrent tasks. But, the product of CODARTS emphasizes only the static aspect of the target system. Therefore, when we perform performance analysis after designing a system, it is

possible to derive the better real time S/W analysis&design by transforming into Petri Net for emphasizing the dynamic aspect between the system's component.

1. 서 론

현실 세계에서는 무수히 많은 현상들이 동시 다발적으로 발생한다. 이러한 동시 다발적인 현상을 가지는 시스템을 모델링하기란 매우 어렵다. 이러한 것들을 시스템으로 모델링하기 위해 많은 학자들이 연구를 하였다. 그 중에 CODARTS(COncurrent Design Approach for Real-Time Systems)[1]와 Petri Net[2]이 실시간 병렬시스템 모델링에 많은 공헌을 하였다. CODARTS는 정적인 측면만을 강조하여 동적인 측면의 지원을 할 수가 없는 단점이 있다. Petri Net은 정적인 측면뿐만 아니라 모델을 실행함으로서, 동적인 측면을 잘 표현하고 있어서

실시간 병렬 시스템의 모델 및 분석에 자주 활용되고 있다.

이 논문에서는 CODARTS 요소 중 TAD (Task Architecture Diagram)를 Petri Net으로 변환시켜서 CODARTS의 정적인 특징에 Petri Net의 동적인 특징을 추가하여 실시간 시스템의 성능 분석을 좀더 효율적으로 하고자 한다. CODARTS로 설계된 모델을 분석하기 위해, Petri Net으로 변형을 한 뒤, 수행능력 검사뿐만 아니라, Petri Net의 기본적인 성질 즉, Liveness, Safeness, Deadlock 검사, Bounded 등 여러 가지 성질을 조사하여 모델의 성능 및 분석을 조직적으로 할 수 있다.

2장에서는 CODARTS와 Petri Net의 개념을 정립하고, 3장에서는 변환을 위한 규칙을 정의한다. 4장에서는 예제를 보여주고, 5장에서 결론 및 향후 연구 방향을 제시한다.

2. CODARTS와 Petri Net의 개요

2.1 CODARTS 개요

동시작업으로 이루어진 system을 설계할 때 사용되는 방법인 CODARTS는 1980년대 초에 제안된 뒤, DART, DARTS/DA, ADARTS, CODARTS 순으로 발전했다. [표 1]에 각 방법에 대한 특징을 기술하였다[1].

2.2 Petri Net의 개요

1964년 독일 Bonn 대학의 C.A.Petri의 박사 학위 논문으로 발표된 페트리 넷은 플레이스, 트랜지션, 초기 토큰 분포 및 각 플레이스와 트랜지션간의 입 · 출력 함수(화살표)로 구성되고 임의의 정점을 여러 개의 호로 나눌 수 있는 방향성 그래프로 정의된다. 임의의 시스템을 Petri Net으로 모델을 하고난 뒤, Petri Net을 분석함으로써 그 시스템의 구조와 동적인 행위에 대한 중요한 정보를 찾아낸다. 이 정보는 모델된 시스템을 평가하는데 사용된다. Petri Net의 특징은 Enable, Firing, Reachable, Deadlock, Safe, Bounded 등이 있다. [표 2]에서 Petri Net의 여러 가지 특징을 정리하였다[3][4][5].

Petri Net의 구조 C는 4개의 tuple을 가진다. 즉, $C = (P, T, I, O)$이다. $P = \{p_1, p_2, \ldots, p_n\}$는 $n \geq 0$인 플레이스들의 유한집합(finite set)이다.

방법론	특징
DART [Design Approach for Real-Time Systems]	여러 개의 동시 태스크로 이루어지는 시스템 개발을 고려한다.
DARTS/DA [DARTS for Distributed Applications]	분산 실시간 응용프로그램의 개발을 고려한다.
ADARTS [Ada-based Design Approach for Real-Time Systems]	Ada 비반 동시 태스크, 실시간 시스템의 개발을 고려하고, DARTS보다 좀더 유지 보수가 쉽고, 재사용 가능한 설계를 제공한다.
CODARTS [COncurrent Design Approach for Real-Time Systems]	특정 언어 지향적인 설계 방법론이 아니고, COBRA(Concurrent Object-Based Real-Time Analysis) 접근법을 제공한다. 또한, 분산 응용프로그램의 설계를 지원한다.

[표 1] CODARTS의 변천사

$T = \{t_1, t_2, \ldots, t_m\}$은 $m \geq 0$인 트랜지션의 유한집합(finite set)이다. 플레이스의 집합과 트랜지션의 집합은 $P \cap T = \emptyset$로 disjoint된다. I: $T \rightarrow P^{\infty}$는 트랜지션으로부터 플레이스의 bag(bag of place)으로 사상되는 입력함수이다. O: $T \rightarrow P^{\infty}$는 트랜지션으로부터 플레이스의 bag(bag of place)으로 사상되는 출력함수이다.

3. CODARTS의 TAD로부터 Petri Net으로의 변환

3.1 Petri Net으로 전이를 위한 CODARTS의 요소

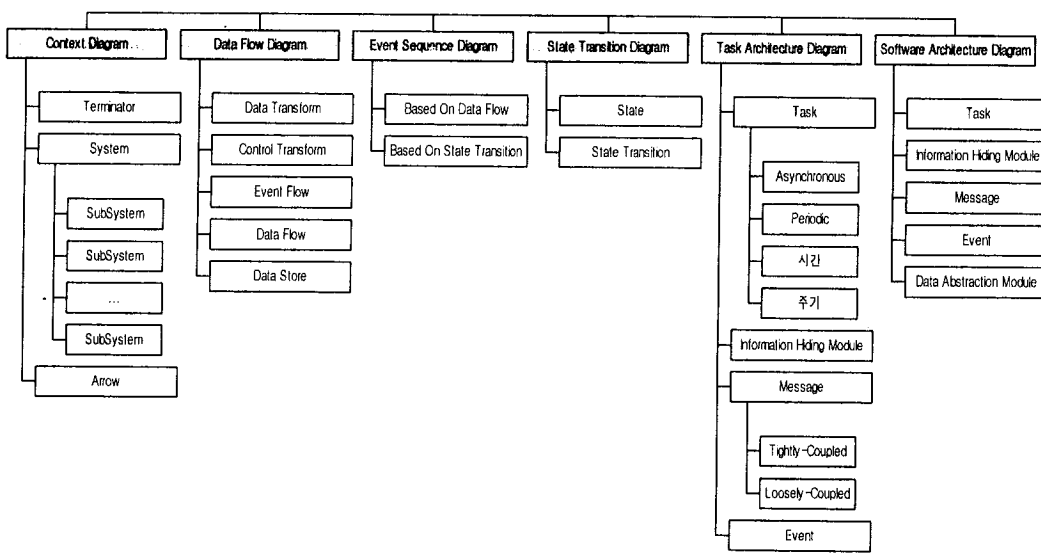
[표 1]과 같은 변천사를 가진 CODARTS에서 Petri Net으로 변환하기 위해 CODARTS를 구성하는 요소를 추출한다.

추출되는 요소는 [그림 1]에서 요약한다. 출되는 요소는 문맥 다이어그램(context diagram), 데이터플로우 다이어그램(data flow diagram), 연속 이벤트 다이어그램(event sequence diagram), 상태전이 다이어그램(state transition diagram), 태스크 구성 다이어그램(task architecture diagram), 소프트웨어 구성 다이어그램(software architecture diagram)으로 구성된다. 문맥 다이어그램은 종결자(Terminator), 시스템(System), 화살표(Arrow)로 구성이 되고, 데이터플로우 다이어그램은 데이터 변형(Data transform), 이벤트 흐름(Event Flow), 데이터 저장소(Data Store)로 구성이 된다.

속성	특징
Enable	임의의 트랜지션의 입력 플레이스들 모두에 토큰이 존재할 때를 나타낸다.
Firing	트랜지션의 입력 플레이스에 있던 토큰들이 1개씩 출력 플레이스로 옮겨지는 것을 나타낸다.
Reachable	초기 토큰 분포로부터 일정한 순서로 점화가 계속되어 원하는 토큰 분포로 될 수 있다는 것을 나타낸다.
Deadlock	PN내의 점화되지 못하는 트랜지션이 있을 때, 그 PN은 deadlock이 발생할 수 있다는 것을 의미한다.
Safe	모든 플레이스마다 토큰이 1개까지만 존재할 수 있다는 것을 나타낸다.
Bounded	토큰의 수가 무한히 증가하지 않는 것을 나타낸다.

[표 2] Petri Net의 특징

연속 이벤트 다이어그램은 데이터플로우 다이어그램과 상태전이 다이어그램의 내용을 기본으로 하여 구성된다. 상태전이 다이어그램은 상태(State)와 상태전이(State Transition)로 구성이 되고, 태스크 구성 다이어그램은 태스크(task),이벤트(event), 메시지(message), 정보은닉모듈(information hiding module)로 구성이 된다. 소프트웨어 구성 다이어그램(Software Architecture Diagram)은 태스크(Task), 정보은닉모듈(Information Hiding Module), 메시지(Message), 이벤트(Event), 데이터추상 모듈(Data Abstraction Module)로 구성이 된다. 이 논문에서는 위의 구성요소 중 태스크 구성 다이어그램으로부터 Petri Net으로의 변형을 시도한다. 그리고, 나머지 다이어그램은 변형을 시도할 때 필요한 정보를 제공한다.

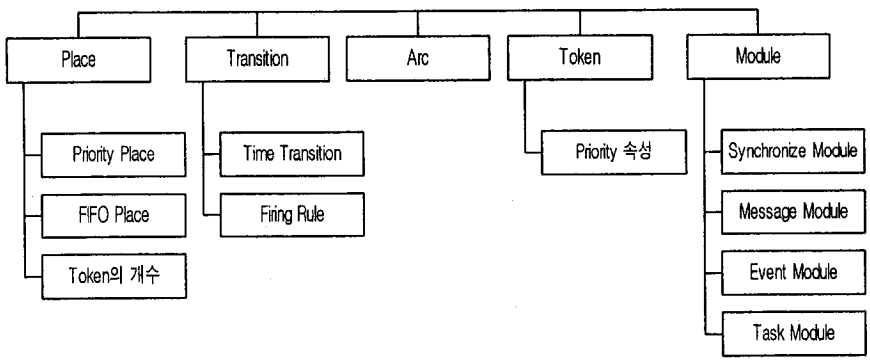


<그림 1> 추출될 CODARTS의 요소

태스크 구성 다이어그램의 태스크는 비동기(Asynchronous) 태스크와 주기적(Periodic) 태스크로 나누고, 추가로 시간과 주기를 표기해 준다. 시간과 주기는 Petri Net이 시뮬레이션하기 위한 필수적인 요소이다. 그리고 메시지 통신의 경우, loosely coupled message와 tightly coupled message로 나눈다. 공유된 데이터를 접근하기 위해, 정보은닉모듈을 사용하고, 한 개 이상의 태스크에 의해 접근된다. 그래서, 이 정보은닉모듈에 접근하는 태스크는 동기화가 된다.

3.2 CODARTS의 요소와 연결되는 Petri Net 요소의 추출

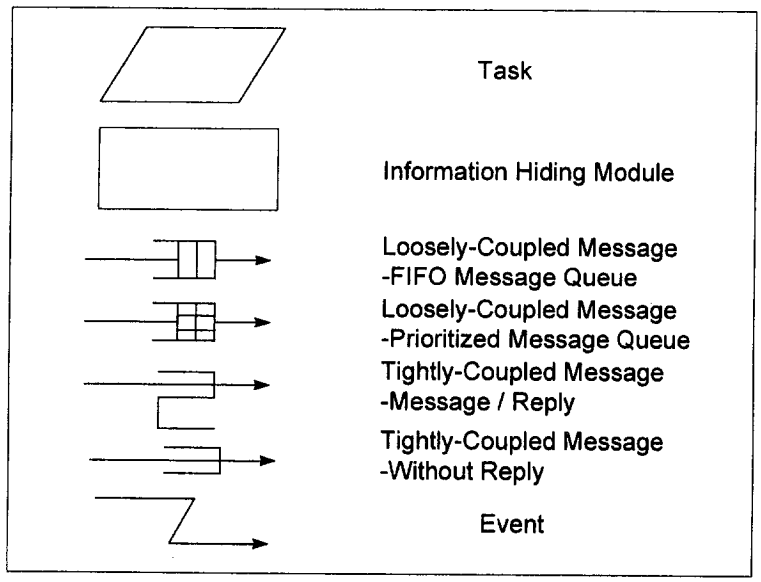
Petri Net의 관점에서 CODARTS 요소와 연결을 시켜야 한다. 추출된 Petri Net 요소는 [그림 2]에서 보여준다. 추출된 요소는 플레이스(Place), 트랜지션(Transition), 호(Arc), 토큰(Token), 모듈(Module)로 나누어진다.



<그림 2> 추출될 Petri Net의 요소

플레이스는 우선순위 플레이스(Priority Place)와 FIFO 플레이스로 구성이 되고 토큰의 개수로 구성된다. 트랜지션은 주로 시간 트랜지션(time transition)과 즉시 트랜지션(immediate transition)으로 구성이 되고, Firing 규칙이 정의가 된다. 토큰은 우선순위 플레이스이기 때문에 토큰 자체가 우선순위 특징을 가져야 하고, 모듈은 플레이스, 트랜지션과 토큰들의 조합으로 구성이 된다. 모듈은 일치(Synchronize) 모듈, 메시지 모듈, 이벤트 모듈, 태스크 모듈로 구성된다.

3.3 Petri Net으로의 변환을 위한 CODARTS의 정의



<그림 3> Task Architecture Diagram Notation

3.1절에서도 언급을 했듯이, CODARTS의 구성 요소 중 태스크 구성 다이어그램을 Petri Net으로 변환을 한다. 태스크 구성 다이어그램의 상세한 구성 요소는 태스크, 정보은닉 모듈, Loosely-Coupled Message Communication-FIFO Message Queue, Loosely-Coupled Message Communication-prioritized Message Queue, Tightly-Coupled Message Communication-Message With Reply, Tightly-Coupled Message Communication without Reply, 이벤트로 구성된다. 태스크 구성 다이어그램의 구성 요소에 대한 다이어그램 표기법은 [그림 3]에서 기술된다.

[그림 3]에서 설명되는 태스크는 비동기 태스크와 주기적 태스크로 나눈다. 비동기 태스크는는 요구될 때 실행되고, 주기적 태스크는 임의의 주기 내에 한 번씩 실행을 하는 특징을 가진다.

Petri Net으로 바꾸기 위해서는 좀더 상세한 정보가 추가된다. 모델링된 다이어그램에서 각각의 태스크 수행 시간(C)과 주기(T)를 표현하고, 그 태스크가 주기적인지 비동기인가를 표현한다. 이러한 정보는 Petri Net이 실행할 때, 트랜지션의 firing 시간과 밀접한 관련이 있다.

3.4 변형된 Petri Net의 정의

이미 2.2절에서 Petri Net의 기본적인 정의를 하였다. 이 절에서는 3.3절에서 설명한 CODARTS를 수용할 수 있는 Petri Net를 정의한다. 이 Petri Net의 정의에는 다음 사항이 포함된다.

1) 트랜지션

수행능력을 조사하기 위해서, 시간 개념이 존재하는 시간 트랜지션을 정의한다[6]. 또한, CODARTS의 태스크 자체가 주기성을 가지기 때문에, 이러한 성질을 나타내기 위해 트랜지션이 주기의 성질을 가져야 한다. 시간이 표시되지 않은 태스크는 즉시 트랜지션으로 해석되는 C=0으로 간주하거나 default 시간(D=상수)으로 간주한다.

속성) 비동기 태스크가 Petri Net으로 바뀔 때, $C_i = \tau_i$이다. (C_i: 태스크 수행 시간, τ_i: 트랜지션의 지연 시간) 트랜지션의 지연 시간 τ_i로만 표현되는 태스크 구성 다이어그램의 요소는 비동기 태스크와 이벤트이다.[그림 4.(a),(d)]

속성) 주기적 태스크가 Petri Net으로 바뀔 때, 주기 T를 정의한다. 트랜지션의 지연 시간 τ_i와 주기 T_i로 표현되는 태스크 구성 다이어그램의 요소는 주기적 태스크와 이벤트이다[그림 4.(b)].

2) 플레이스와 토큰

플레이스는 큐를 표현할 수 있어야 하기 때문에, 1개 이상의 토큰을 가질 수 있는 특징을 가진다. 또한, 토큰 자체의 우선순위 속성 때문에, 우선순위 순으로 토큰을 트랜지션으로 전달을 할 수 있어야 한다. 또한, FIFO의 특성을 가지기 위해, 플레이스로 입력되는 토큰에 순번의 속성을 조절할 수 있는 플레이스를 제안한다[3].

속성) Petri Net내의 임의의 플레이스 P_i에 대해, $cap(p_i)$은 p_i내의 토큰의 허용 개수[3]를 표현한다.

속성) 토큰은 우선순위(token.priority), 순위(token.order)와 같은 속성을 가진다.

속성) 태스크 구성 다이어그램의 메시지는 Petri Net에서 플레이스, 토큰과 트랜지션의 조합으로 나타난다. 플레이스는 토큰의 개수가 1개 이상이다. 즉, $cap(p_i) \geq 1$이다. 여기에 속하는 태스크 구성 다이어그램의 요소는 Tightly Coupled Message with Reply, Loosely-Coupled Message-FIFO, Loosely-Coupled Message Queue-Priority, Tightly-Coupled Message Communication without Reply이다.

속성) 태스크 구성 다이어그램의 메시지는 FIFO 기능을 가지는 플레이스로 표현된다. 이 플레이스는 토큰의 속성 중 token.order를 사용해 순서대로 내보낼 수 있는 특징을 가진다. 여기에 속하는 태스크 구성 다이어그램의 요소는 Loosely-Coupled Message-FIFO이다[그림 4.(c)].

속성) 태스크 구성 다이어그램의 메시지는 우선순위를 기능을 가지는 플레이스로 표현

된다. 이 플레이스는 토큰의 속성 중 token.priority 속성을 사용해 토큰의 우선순위 순으로 출력한다. 여기에 속하는 태스크 구성 다이어그램의 구성 요소는 Loosely-Coupled Message Queue Priority이다[3][그림 4.(e)].

3) 초기 마킹(Initial Marking)

속성) 초기 마킹이란 시스템이 처음 시작할 때의 상태를 말하고, Petri Net의 초기 토큰 분포를 의미한다.

Petri Net의 초기 마킹은 정보은닉모듈의 ACK, Tightly-Coupled Message Communication without Reply의 ACK, Tightly-Coupled Message With Reply의 ACK에 토큰이 놓인다.

4) 일치 모듈(Synchronization Module)

태스크 구성 다이어그램의 정보은닉모듈은 여러 태스크가 동시에 접근을 하기 때문에, 동시 접근을 제어할 수 있는 Petri Net 일치 모듈이 존재해야 한다. 그래서, 태스크 구성

Component(요소)	CODARTS	Petri Net
Task	Task(Asynchronous)	시간 Transition
	Task(Periodic)	시간+주기 Transition
Information Hiding Module	Information Hiding Module	Petri Net module(synchronize)
Message	Loosely-Coupled-FIFO Queue	FIFO의 특성을 가지는 Place
	Loosely-Coupled-Priority Queue	Priority의 특성을 가지는 Place
	Tightly-Coupled-With Reply	Place와 Transition의 조합
	Tightly-Coupled-Without Reply	Place와 Transition의 조합
Event	Event	Place와 Transition의 조합

[표 3]. 각각의 변이 관계

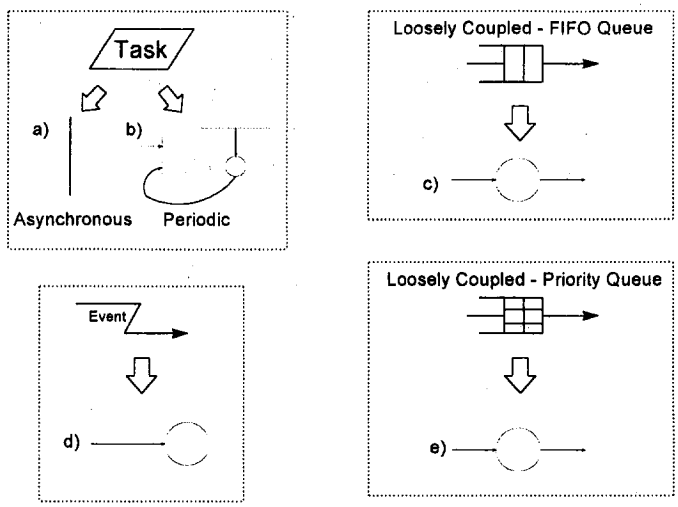
다이어그램의 정보은닉모듈은 일치모듈로 바뀌게 된다.

속성) 일치 모듈에는 단 한 개의 토큰만이 들어갈 수 있다. 즉, 일치 모듈은 $cap(p_i)=1$이다.

5) Petri Net의 분석

일반적으로, Petri Net의 속성인 Reachability, Liveness, Safeness, Deadlock, Bounded를 조사하고, 일정시간 내에 특정 상태가 존재하는 지를 조사해야 한다. 즉, 태스크가 일정시간 이내에 수행이 되는가를 조사해야 한다. 이러한 성질들이 만족하지 못한다면, 태스크 구성 다이어그램의 설계가 잘못 되었기 때문에, 이를 수정해야 한다.

분석 방법[4]은 크게 Reachability Tree 방법과 Matrix Equation 방법인 2가지로 나눈다. Reachability 방법은 Petri Net이 계속 firing을 하여, 토큰의 분포 상태를 tree형태로



<그림 4> TAD에서 Peri Net으로 변환 규칙

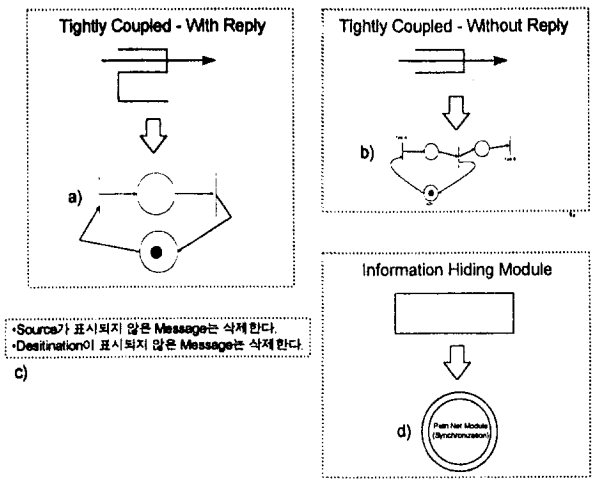
나타나는 방법을 의미하고, Matrix Equation 방법은 Petri Net을 행열 D로 표시한 후, $M_0 = M_0 + D^*$의 해를 구하는 방법이다. $D = D^+ - D^-$이고, D^+는 트랜지션들로부터 출력되는 플레이스의 수를 나타낸 행열이고, D^-는 입력되는 플레이스의 수를 나타낸 행열이다. M_0와 M은 각각 초기 토큰 분포 벡터와 원하는토큰 분포 벡터이다. 해가 존재하면, 그 Petri Net은 Reachability가 있는 것이고, 없으면 Reachability가 없는 것이다.

3.5 TAD의 Petri Net으로의 변환 규칙의 정의

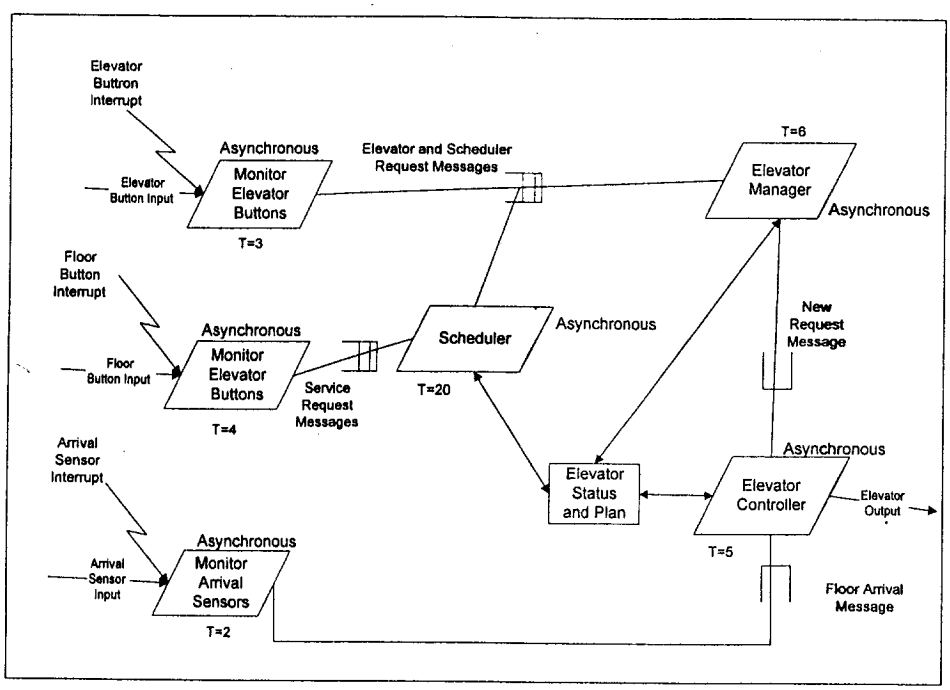
앞절에서 정의한 내용을 바탕으로 CODARTS의 태스크 구성 다이어그램 요소와 Petri Net의 연결 관계를 [표 3]에서 나타내었다. 태스크는 시간 트랜지션(비동기 태스크)과 시간+주기 트랜지션(주기적 태스크)으로 변환되고, 정보 은닉모듈은 일치 역할을 하는 Petri Net 모듈로 변환된다. 메시지는 각각 FIFO, 우선순위의 특징을 가지는 플레이스와 시간 트랜지션의 조합으로 변환된다. 이벤트 역시 주기적으로 토큰을 발생시키는 플레이스와 시간 트랜지션의 조합으로 변환된다.

[표 3]에서 나타난 것을 도식화하여 나타내면 [그림 4], [그림 5]와 같다. [그림 4]에서 태스크는 비동기 태스크와 주기적 태스크로 나누어진다. 비동기 태스크의 수행 시간 C가 트랜지션의 지연 시간 τ와 대응이 된다. 주기적 태스크일 경우, 주기 T마다 토큰을 1개씩 firing해주는 모듈(트랜지션과 플레이스로 구성)로 대응된다. FIFO 방식의 Loosely-Coupled Message경우, FIFO의 특징을 가지는 플레이스로 대응된다[그림 4.(c)]. 우선순위 큐 방식의 Loosely-Coupled Message일 경우, 연결되는 태스크에서 우선순위 속성을 가지는 토큰을 생성하고, 우선순위 토큰을 처리하는 플레이스로 대응된다. 이벤트는 임의의 시간 C마다 토큰을 한 개씩 생성해 내는 모듈로 구성된다. 모듈내의 주기는 사용

자 정의값에 따른다[그림 4.(d)]. Tightly Coupled Message -With Replay는 메시지를 받고 일을 처리할 때까지는 source 트랜지션이 firing을 하지 못하는 모듈로 구성된다[그림 5.(a)]. Tightly Coupled Message Without Reply는 메시지를 받았다는 ACK만 source 트랜지션으로 보내고 난 뒤, 자신의 일을 처리하는 모듈로 구성된다[그림 5.(b)]. 정보은닉 모듈은 여러 개의 트랜지션을 일치시키기 위한 module이다[그림 5.(d)]. 그리고, 변환을 할 때 데스크의 source가 없는 메시지와 Destination 이 없는 메시지는 삭제한다[그림 5.(c)].



<그림 5> TAD에서 Petri Net으로 변환 규칙

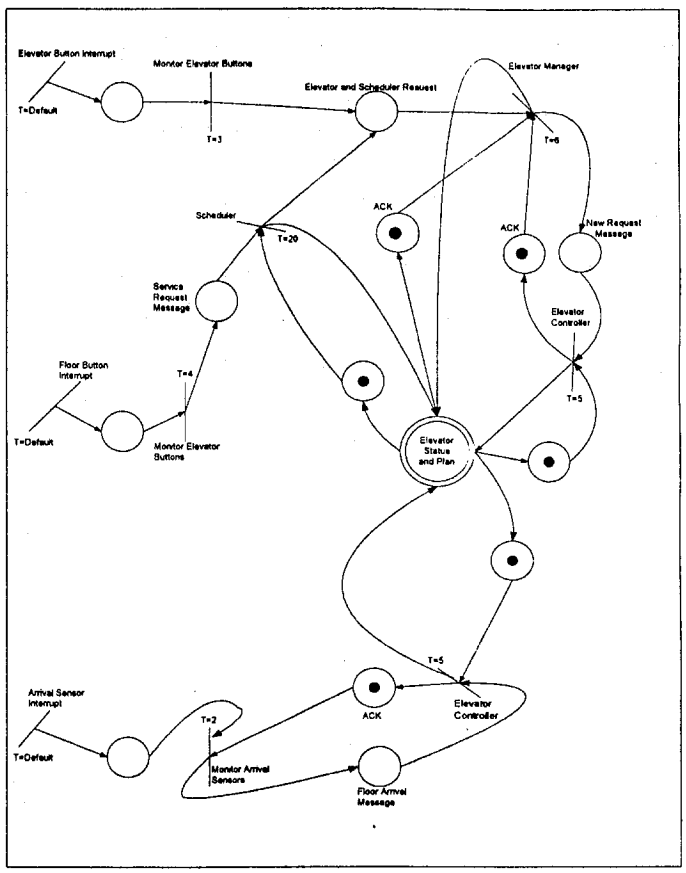


<그림 6> Elevator System Task Architecture Diagram

4. 변환 규칙의 적용 예제

본 논문에서 제안한 변환규칙은 Gomma[1]책의 엘리베이터 시스템의 태스크 구성 다이어그램을 예로하여 적용해 보았다. 이 태스크 구성 다이어그램[그림 6]을 [그림 4]와 [그림 5]에 적용시켜 Petri Net으로 바꾼 결과는 [그림 7]과 같다. [그림 6]의 태스크에는 각 태스크의 수행 시간과 주기가 기술된다. 이 예제에서는 주기가 나타나지 않는다.

분석은 [그림 7]에 나타난 Petri Net을 3절의 Petri Net 분석 방법[4]을 이용한다. 수행 결과로서 나타나는 질의가 몇 가지 존재한다. "이 Net가 계속 살아 움직일 수 있는가?(liveness)", "임의의 부분에 부하가 걸리거나 그 부분에서 멈추는가?(deadlock)", "도달할 수 없는 곳(상태)가 존재하는가?(reachable)", "원하는 시간 이내에 임의의 상태로 갈 수 있는가?(수행능력)"등의 문제를 검토할 수 있다.



<그림 7> Petri Net으로의 변형된 결과

5. 결론 및 향후 연구

위의 예에서 보았듯이, 실시간 소프트웨어 설계 방법인 CODARTS의 정적인 측면에 실시간 소프트웨어의 동적인 측면을 강조하는 Petri Net으로의 변환을 함으로써 좀 더 나은 실시간 소프트웨어의 설계 및 분석을 유도할 수 있었다. 즉, 태스크 구성 다이어그램로 시스템을 설계한 뒤, 좀 더 동적인 특징을 가지는 Petri Net으로 바꿈으로서 Petri Net의 특징인 Liveness, deadlock, safeness 등 여러 가지 특징을 조사할 수 있게 된다. 이러한 조사를 통해 좀더 나은 태스크 구성 다이어그램의 분석을 할 수가 있다. 또한, 여러 가지의 변환 규칙을 제안함으로써, 좀더 체계성을 갖출 수가 있었다.

이후의 연구 과제는 좀더 정량적인 변환 규칙을 제안하고, 이러한 변형을 자동적으로 할 수 있는 도구와 분석을 자동적으로 할 수 있는 도구를 개발할 것이다.

참 고 문 헌

[1] "Real-Time Structured Methods", Keith Edwards, Wiley series in software engineering practice.

[2] "Software Design Methods for Concurrent and Real-Time Systems", Hassan Gomaa, Addison-Wesley Publishing Company.

[3] "페트리네트의 분류법(A Taxonomy of Petri Nets), 이 강수, 한국정보과학회 논문지, 1994.8.

[4] "페트리네트에 관한 기술 해설", 이 강수, 한국정보과학회 논문지, 1983. 6

[5] "Petri Nets-an introduction", W. Reisig, Springer-Verlag, 1985

[6] "Timed Petri Nets definitions, properties, and applications", W.M.Zuberak, Microelectronics and Reliability, Vol.31, No, 4, pp. 627-644, 1991