봉쇄와 교착이 존재하는 시스템의 성능분석을 위한 페트리-네트 기반 모의실험 소프트웨어 개발

Development of a Petri-net-based Simulation Software for Performance Evaluation of the System with Blocking and Deadlock

박찬우*, 황상철*, 이효성*
Chan-Woo Park, Sang-Chel Hwang, Hyo-Seong Lee

Abstract

In this paper, a new software package for modeling and simulating discrete-event dynamic systems is developed. The new software is a general-purpose, graphical tool based on timed Petri-nets and is developed using Visual Basic and Visual C++ for the window environment. It allows the user to graphically build a Petri-net model and enter input data for executing the Petri-net simulation model. It is equipped with a deadlock detection and recovery function as well as an automatic error check function. In addition, the software supports various enabling functions and distribution functions and provides various statistics for the performance measures of interests pertaining to the system. We expect the new software will be used in a wide number of applications including computer, communication and manufacturing systems.

* 경희대학교 산업공학과

1. 서론

지난 십 수년간 통신시스템의 급속한 발전과 제조시스템의 생산환경 변화로 인해 이산형 사건 동적 시스템(discrete event dynamic system)의 중요성에 대한 인식이 더해가고 있다. 이산형 사건 동적 시스템은 이산적 형태로 도래하는 사건에 의해 시스템의 상태가 변화하는 사건구동형 시스템(event-driven system)으로 시간의 추이에 따라 시스템의 상태가 연속적으로 변화하는 시간구동형 시스템(time-driven)과는 특성상 근본적인 차이가 있다[1]. 따라서 시간구동형 시스템에 사용되는 기법을 사건구동형 시스템에는 적용할 수 없으며, 이러한 이유로 사건 구동형 시스템에도 시간구동형 시스템에 준하는 체계적인 성능평가 기법의 개발이 요구된다.

통신시스템, 제조시스템 등 이산형 사건 동적 시스템의 성능평가를 위한 대표적인 기법으로는 수리적 기법과 모의실험기법을 들 수 있다[1]. 대기행렬기법으로 대표되는 수리적 기법은 시스템을 구성하는 관련요소들 간의 상호관계 규명을 가능하게 하고 시스템의 성능 평가치를 신속하고 정확하게 제공해 준다는 점에서 이상적인 기법이라 판단되나 극히 단순한 시스템을 제외하고는 적용이 어려워 사용범위가 크게 제한되어 있다. 반면에 모의실험기법은 사용범위의 제한은 없으나 프로그램 개발에 많은 시간과 비용이 소요되며, 프로그램이 개발된 후에도 시스템의 중요 특성들이 바뀔 때마다 모형의 변형 및 이에 따른 대대적인 프로그램 수정작업이 필요해 소프트웨어 유지비용이 과다하게 소요된다. 이러한 문제점을 해결하기 위한 한가지 대안이 페트리-네트를 이용한 모의실험이다.

페트리-네트는 이산형 사건 동적 시스템을 모형화하기 위한 유용한 도구로서 제조 시스템, 컴퓨터 및 통신 시스템 등을 모형화 하는데 널리 사용되고 있다. 페트리 네트는 우선성(priorities), 동기성(synchronization), 봉쇄(blocking) 등와 같은 비승법형태(non-product form)를 갖는 시스템의 모형화에 특히 유용하게 이용되어 질 수 있으며 다음과 같은 장점을 가지고 있다[1,12]. 첫째, 시스템을 정확하고 용이하게 모형화할 수 있으며, 시스템의 논리적특성의 규명을 가능하게 한다. 둘째, 추계적 페트리 네트 등으로 표현 가능한 일부 시스템의 경우에는 페트리 네트에 내재되어있는 marking process를 분석함으로써 시스템의 수리적 성능분석을 수행할 수 있다. 셋째, 수리적분석이 불가능한 경우에도 페트리 네트 모형을 이용해 모의실험코드를 자동적으로 생성할 수 있으므로 모의실험을 통한 시스템의 성능분석을 가능하게 해준다.

페트리 네트는 개체 모수가 place와 토큰(token)으로 주어지며, 상태의 변화나 전개는 transition으로 주어져 상호 연관관계를 arc로 표현하기 때문에 대상 시스템의 변화를 해당 토큰, place 와 transition 만을 추가 혹은 삭제함으로써 시스템의 특성 변화에 따른 모형 수정이 극히 용이하다. 따라서, 페트리 네트의 가장 큰 장점인 사용자의 편의성, 도식성 등을 이용하여 시스템을 모델링하고, 페트리 네트의 특성들로부터 모의실험코드를 자동생성한다면 모의실험에 들어가는 초기개발비용 및 유지비용을 크게 줄일 수 있게 된다. 이러한 이유로 외국에서는 Chiola[2], Ciardo[3] 등에 의해 개발이 시작된 이후 이미 여러 소프트웨어들이 개발된 상태이다[7, 8, 11]. 그러나 이들 소프트웨어들은 사용자 편의성과 다양한 기능 제공이라는 측면에서 보아 아직 완성된 수준에 이르지 못하고 있으며 보완되어져야만 할 측면이 상당부분 존재한다. 특히 기존의 소프트웨어에서는 모의실험 도중에 교착(deadlock)이 발생하는 경우에 이를 탐지하고 해소하는 기능을 갖고있지 못하여, 교착이 발생하는 시스템을 모의실험할 경우에는 사용자가 이를 직접 코딩하여야하는 불편이 따른다.

기존의 소프트웨어의 이와 같은 단점을 보완하기 위해 본 연구에서는 교착 탐지 및 교착해소 기능을 갖는 시뮬레이션 소프트웨어를 페트리-네트를 이용하여 개발하였다. 본 소프트웨어는 교착탐지 및 교착해소 기능이외에도 사용자의 편의성 측면과 기능의 다양성 측면에서 기존의 소프

트웨어에 비해 우수한 것으로 평가된다. 실사회에는 봉쇄와 교착이 존재하는 시스템이 많고, 본 소프트웨어는 비전문가라도 쉽게 사용할 수 있도록 사용자 편의성이 충분히 고려되었으므로 본 소프트웨어의 실용적 가치는 클 것으로 기대된다.

2. 시뮬레이션 S/W의 설계 및 구현

본 연구에서는 사용자가 시간형 페트리-네트를 이용하여 대상 시스템을 그래프 형태로 표현할 수 있도록 윈도우(window) 환경 하에서 Visual Basic[9]와 Visual C++ [10]을 이용하여 개발하였으며, 사용자의 편의성 제고와 기능의 다양성을 소프트웨어 개발의 최우선 목표로 두었다. 이를 위하여 그래픽 기능의 활용을 극대화하고, 페트리-네트의 기본 모형인 place, transition, arc 기호의 다변화를 통하여 시스템 기능의 다양성 및 모형화 능력을 높이고자 하였다.

개발된 소프트웨어는 사용자가 이산형 사건 동적 시스템을 그래픽도구를 이용하여 페트리-네트로 표현하여 주면 모의실험을 자동으로 수행하여 시스템의 중요한 성능 분석치를 제공하여 주며, 소프트웨어의 주요 기능 및 특성은 다음과 같다.

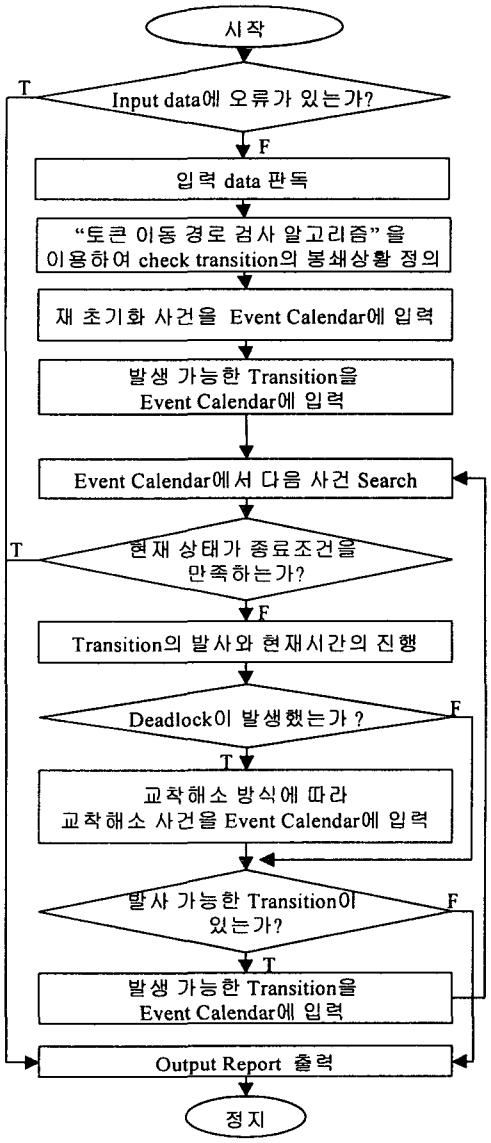
첫째, 그래픽 도구를 이용한 시스템 모델링 기능, 자동 error-check 기능 등 사용자 편의성 및 소프트웨어의 유연성을 극대화하고자 하였다.

둘째, 본 연구에서는 double linked-allocation approach기법을 사용하였다. 물리적으로 인접한 각각의 record를 저장하는 sequential-allocation approach기법과 달리 double linked-allocation approach기법에서는 각각의 record가 상호 논리적 관계를 갖게되므로 이를 이용하면 event calendar의 관리가 용이해 진다[5]. 따라서 모의실험에 소요되는 시간과 컴퓨터 메모리의 사용량이 감소될 수 있으며, 대규모 event record를 필요로 하는 모의실험의 경우 특히 효과적일 것으로 기대 된다.

셋째, 교착 탐지기능과 교착해소 방식의 도입이다. 서비스를 끝낸 두 개 이상의 unit에 상호 봉쇄(blocking)현상이 발생하여 시스템이 교착상태에 빠질 경우, 이를 즉시 탐지하고 교착을 해소해 주는 기능을 모의실험 모듈에 포함시켰다. 본 연구에서는 교착의 원인이 되는 transition의 순차적 동시 발사(fire)에 의해 토큰을 목적 place에 이동시킴으로써 교착을 해소하는 방식을 사용하였으며, 교착해소에 소요되는 시간은 무시할 수 있을 정도로 작거나 일정한 시간이 소요될 수 있다고 가정하였다.

넷째, arc의 종류를 다변화하고, 지수, 정규, 일양, 감마, 얼랑, 콕시안 분포 등 다양한 확률분포를 지원하며, place에 토큰의 저장·제거 방법을 다변화함으로써 시스템의 모델링 능력을 높이고자 하였다.

소프트웨어 개발 알고리즘은 <그림 1>과 같다. 시스템 모형화 초기에 사용자는 대상 시스템을 place, transition, arc를 이용하여 표현한다. 이를 표현할 때 네트워크 모형을 도시함과 동시에 프로그램 내부에서는 입력 data에 오류가 있는지를 검사하게 된다. 예를 들면 place와 place 또는 transition 과 transition을 arc로 연결했을 때 자동적으로 오류메시지를 출력한다. 따라서 사용자는 대상시스템을 표현할 때 모형의 오류 또는 입력 데이터 값의 오류가 발생할 경우 즉시 이를 확인할 수 있게 되며, 이 단계는 사용자가 대상시스템을 완성할 때까지 반복된다. 모의실험이 시작되었을 때 모든 변수들과 통계치가 초기화된다. 이 단계 이후 사용자가 입력한 데이터는 프로그램 내부에서 판독된다. 봉쇄상황을 정의하기 위한 사전단계로서 토큰이동 검사 알고리즘을 이용하여 check transition의 blocking상황을 사전에 정의하고 저장한다. 다음 단계에서는 모의실험 종결사건과 재 초기화 사건을 event calendar에 저장한다. 개발된 소프트웨어에서는 모의실험 종결사건과 재 초기화 사건의 발생 조건을 사용자 입력에 따라 두 가지 종류로 나누었다. 첫 번째는 사용자가 지정한 시간에 발생시키는 경우와 두 번째는 사용자가 지정한 특정 transition의 발사 횟수에 따라 종결사건과 재 초



<그림 1> 모의실험 소프트웨어의 흐름도

기화 사건이 발생하는 경우를 말한다. 본 연구에서는 모의실험의 사건(event)을 네 가지 경우로 구분하였다. 첫 번째 사건은 토큰의 이동 조건이 충족될 경우 발생하는 transition 발사 사건이며, 두 번째 사건은 시스템 초기상태의 영향을 감소시키기 위한 모의실험의 재 초기화 사건이다. 세 번째 사건은 교착(deadlock)이 발생할 경우 교착 상태를 해제하기 위한 사건이다. 마지막으로 네 번째 사건은 모의실험 종결 사건이다. 여기서, event calendar란 앞으로 발생한 사건들을 시간의 증가순으로 정렬한 배열을 의미하며, 소프트웨어는 event calendar의 사건의 유형에 따라 각기 다른 방식으로 모의실험을 진행한다. 다음 event time이 되어 transition이 발사되면 이에 따라 토큰의 이동이 이루어지고 새로이 enable된 transition을 event calendar에 포함시킨다. event calendar의 다음 사건이 모의실험 종결사건이면 수집된 통계치를 output-report에 출력한다. <그림 2>은 개발된 소프트웨어의 화면 구성과 실행 화면의 예를 나타낸다.

3. 봉쇄와 교착 탐지 및 교착 해소 알고리듬

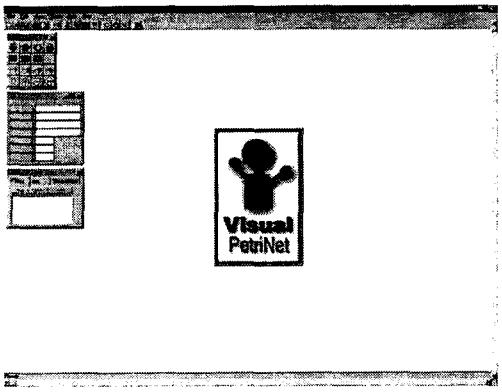
봉쇄(blocked) 및 교착은 모두 노드(node)의 대기공간(buffer size)이 제한되어 있을 때 발생하는 현상이다. 봉쇄는 단순히 노드의 대기공간이 제한되어 있을 때 발생하며 교착은 대기공간이 제한되어 있는 노드간에 feedback loop이 존재할 때 발생한다. 통신 시스템과 자동화된 생산 시스템에서는 노드의 대기공간이 극히 작게 설계되어 있는 경우가 많아 봉쇄와 교착이 빈번히 발생할 가능성이 높다. 따라서 봉쇄 및 교착을 탐지하고 교착을 해소시켜주는 기능을 소프트웨어에 포함시키는 것은 매우 바람직할 것이다.

3.1 기존 페트리-네트에서의 봉쇄의 정의

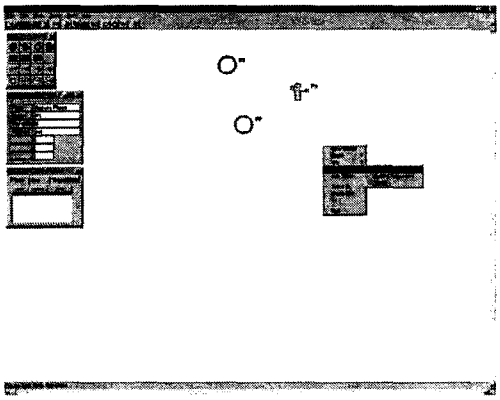
기존의 페트리-네트에서는 다음과 같은 상황에서 현재 marking M에서 transition t가 봉쇄되었다고 정의한다[12].

□ 정의 1

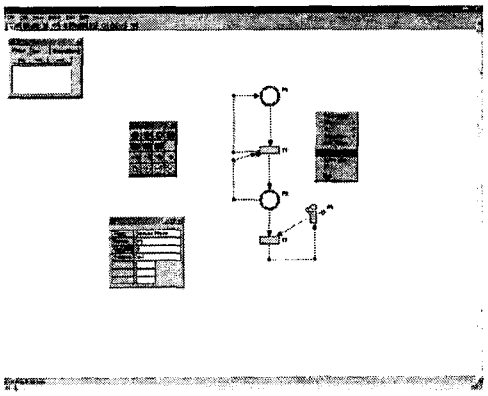
임의의 페트리-네트 모형 (P, T, IN, OUT, M_0)이 주어지고, 현재 marking $M(M \in R(M_0))$에서 임의의 transition t 는 다음 조건을 만족하면 봉쇄되었다고(blocked) 정의한다.



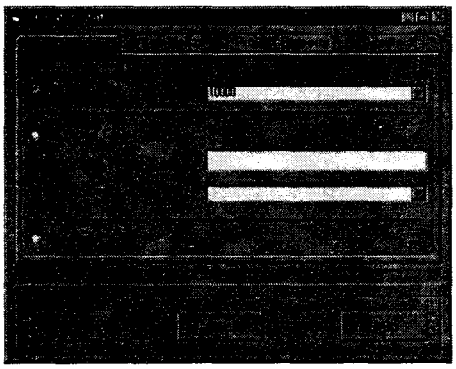
(a) 초기화면



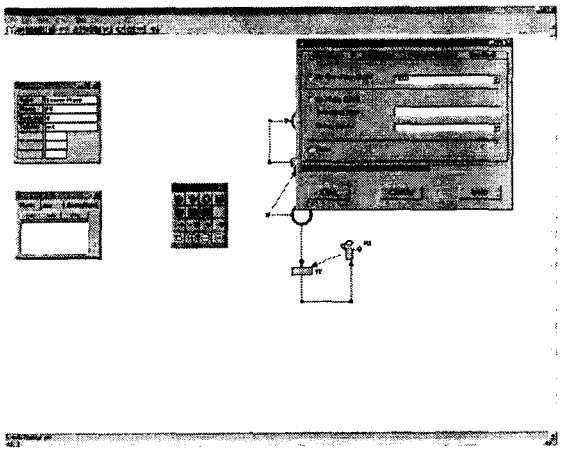
(b) 네트워크



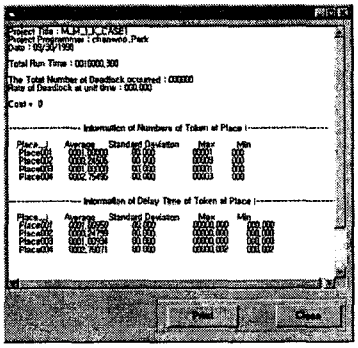
(c) 완성된 네트워크



(d) 프로젝트 자료의 입력



(e) 모의실험 진행화면



(f) output report 출력

<그림 2> 실행화면

(a) t는 두개 이상의 input place를 가지고,
b) $M(p) \geq IN(p, t)$인 place p가 존재하며,
(c) t는 현재 marking M에서 enable이 되지 않을 때.

정의 1에서 보는 바와 같이 현재 marking M 상태에서, transition t의 봉쇄 상황을 정의할 수 있으나, 정확히 transition t가 어떤 transition의해 봉쇄되었다고 정의 할 수 없어, 정확한 봉쇄의 원인을 규명할 수 없다. 따라서 transition t의 봉쇄 상황의 재 정의가 필요하다.

3.2 페트리-네트에서의 봉쇄 재정의

기존 페트리-네트의 기호만으로는 transition t의 input-place 중에서 어떤 place가 정의 1의 (b)를 의미하는 place인지 파악이 불가능하다. 따라서 본 연구에서는 place를 normal, queue, resource, server로 구분하여 사용하도록 한다.

이러한 가정 하에서 blocked 상황이란 "임의의 시스템의 하부 프로세스에서 다른 하나의 하부 프로세스로 이동시 필요로 하는 자원이 없을 경우 발생하고, 봉쇄된 고객은 대기장소에서 필요한 자원이 충족 될 때까지 대기하는 상태"로 해석할 수 있다. 따라서 봉쇄발생 시 하부 프로세스(S_i)는 다른 하부 프로세스(S_j)로 토큰의 이동에 필요한 자원이 충족될 때를 기다리는 "하나"의 대기 장소를 필요로 하며, 이를 Queue Place라 정의한다. Queue Place에서 대기중인 토큰이 S_j로 이동하기 위해서는 하나 이상의 자원을 필요로 하며, 이런 자원이 있는 장소를 resource 또는 server-place라 정의한다.

(a) Normal Place (b) Queue-Place

(c) Resource-Place (d) Server-Place

<그림 3> 확장된 페트리-네트의 기호

다음은 blocking 상황을 정의하기 위해 필요한 매개 변수이다.

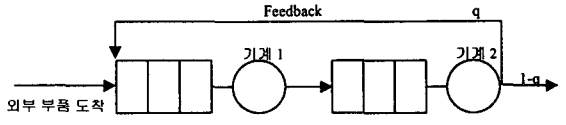
□ 매개변수
- $ct(q_i, l)$: queue-place q_i의 l번째 output transition. 앞으로 $ct(q_i, l)$를 check transition 라 언급한다.
- $ct(q_i, l) \rightarrow ct(q_j, m)$: $ct(q_i, l)$가 $ct(q_j, m)$를 봉쇄한(blocking)경우 이를 $ct(q_i, l) \rightarrow ct(q_j, m)$로 언급한다.
- $RPL[ct(q_i, l)]$: $ct(q_i, l)$의 input place가 되는 resource 또는 server place의 집합.
- $NPL[ct(q_i, l)]$: $ct(q_i, l)$의 input-place중에서 $RPL[ct(q_i, l)]$를 제외한 place의 집합.

앞에서 언급했듯이 blocked 상황이란 "queue place에 있는 토큰이 이동하기 위하여, queue -place의 out-transition t 가 발사가능상태가 되기 위하여 필요한 token이 server 또는 resource place에 있기를 기다리는 상황"으로 정의할 수 있다. 따라서 blocked 상황이 발생했을 경우, 발사(fire)에 의한 토큰의 이동으로 t 를 발사 가능상태로 만들 수 있는 하나 이상의 transition이 존재하며, 이러한 transition사이에서는 다음과 같은 임의의 토큰이동경로(R)가 존재한다.

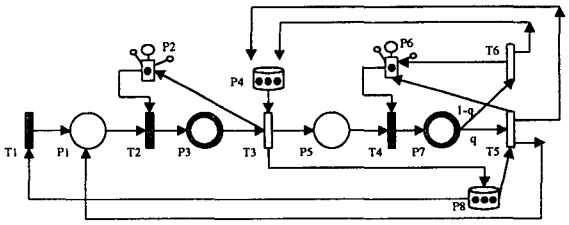
$$R = ct(q_i, l) \rightarrow P_1 \rightarrow \ldots P_n \rightarrow ct(q_j, m)$$

예를 들어 <그림 4>의 2개의 노드로 구성된 대기 시스템을 생각해 보자. 첫 번째 노드에서 서비스 받은 고객은 두 번째 노드에서 서비스를 받아야 한다. 확률 q 로 첫 번째 서비스를 다시 받아야하며, 각각의 노드는 최대 3명의 고객까지

받을 수 있다고 가정하자. 이를 확장된 페트리-네트모형으로 나타내면 <그림 5>과 같으며, 각 기호의 의미는 <표 1>과 같다.



<그림 4> 2개의 노드로 구성된 대기 시스템



<그림 5> 2개의 노드로 구성된 대기 시스템의 확장된 페트리-네트 모형

<표 1> 2개의 노드로 구성된 대기 시스템의 확장된 페트리-네트 모형의 묘사

기 호	의 미
P2	노드 1의 서버.
P3	노드 1에서 서비스를 끝낸 부품이 노드 2에서 서비스 전의 대기 장소.
P4	노드 2의 여유 공간(capacity).
P6	노드 2의 서버.
P7	노드 2에서 서비스를 끝낸 부품이 노드 1에서 서비스전의 대기 장소.
P8	노드 1의 여유 공간(capacity).

<그림 3>에서 {P3, P7}은 queue-place, {P2, P6}은 server-place, 그리고 {P4, P8}은 resource -server를 나타내며, 앞에서 정의된 매개변수의 예는 다음과 같다.

$ct(P3,1) = T3$, $ct(P7,1) = T5$, $ct(P7,2) = T6$
$RPL[T3] = \{P4\}$, $NRPL[T3] = \{P3\}$
$RPL[T5] = \{P8\}$, $NRPL[T5] = \{P7\}$

<그림 3>에서 볼 수 있듯이 다음과 같은 두 개의 자원 이동경로가 존재함을 알 수 있다.

$R_1 = T3 \rightarrow P8 \rightarrow T5$, $R_2 = T5 \rightarrow P4 \rightarrow T3$

이러한 blocked 상황을 정의하는 이동 경로 R은 다음의 존재 조건을 만족하여야 한다.

■ 봉쇄를 정의하는 토큰 이동경로 존재 조건:

존재 조건 1

이동경로 R은 $ct(q_i, l)$을 시작으로 $ct(q_j, m)$까지 이동경로를 완성하는 place와 transition의 arc의 상호 연결이 있어야 하며, 이동경로 상에 이미 존재하는 place와 transition은 한번 이상 같은 경로에 포함될 수 없다.

$ct(q_i, l)$이 $ct(q_j, m)$의 blocked 상황을 정의하기 위해서는 이동경로를 완성하는 경로가 존재하여야 하며, 한번 이동경로에 포함된 place, transition이 다시 이동경로에 존재하는 경우 self-loop가 발생하여 $ct(q_j, m)$의 blocked 상황을 정의하지 못한다.

존재조건 2

$ct(q_j, m)$의 input-place에는 반드시 queue -place가 있어야 하여, resource 또는 server place가 적어도 한 개 이상 존재하여야 하며, 이 동경로의 마지막 place는 resource 또는 server place이어야 한다.

위의 조건을 만족하는 이동경로 R이 하나 이상 존재하는 경우, 시스템 상에는 봉쇄가 존재할 수 있으며 봉쇄를 정의하는 토큰 이동조건을 이용하면 다음의 정리가 성립한다.

■ **정리 1**

임의의 페트리-네트 모형 (P, T, IN, OUT, M_0)이 주어지고, 현재 marking M ($M \in R(M_0)$)에서 다음의 조건을 만족하면, 임의의 check transition $ct(q_j, m)$는 $ct(q_i, l)$에 의해서 봉쇄되었고(blocked), $ct(q_i, l)$는 $ct(q_j, m)$를 봉쇄한다

(blocking).

(a) $ct(q_i, l)$와 $ct(q_j, m)$는 두개 이상의 input place를 가지고,

(b) $ct(q_i, l)$에서 $ct(q_j, m)$까지 존재 조건 1, 2를 만족하는 하나 이상의 이동경로가 존재하며,

(c) $M(p) \geq IN(p, t)\ \forall\ p$,
$where\ p \in NPL[ct(q_j, m)]$

(d) $M(q) < IN(q, t)\ \exists\ q$,
$where\ q \in RPL[ct(q_j, m)]$

여기서,

$$NPL[ct(q_j, m)] \cap RPL[ct(q_j, m)] = \emptyset,$$
$$NPL[ct(q_j, m)] \cup RPL[ct(q_j, m)] = IN(ct(q_j, m))$$

3.3 토큰 이동검사 알고리즘

본 연구에서는 다음의 알고리즘을 이용하여 transition의 봉쇄상황을 모의실험 초기에 찾아 저장한다. 다음의 알고리즘은 존재조건 1, 2을 만족하는 토큰 이동경로 검사를 수행하여 정리 1의 봉쇄상황을 정의하는 과정을 나타낸다.

□ 토큰 이동검사 알고리즘

단계 1) 모든 경로를 초기화. 시작 Check Transition을 선택하고 경로에 포함. 이 경로를 "New" 로 표시.

단계 2) "New" 로 표시된 경로가 있을 때까지 반복

단계 2.1) "New" 로 표시된 경로 선택(R). 현재 경로를 R로 하여 마지막으로 추가된 transition을 T라 하자.

단계 2.2) T의 output place가 없다면, R을 "End" 로 설정. GO TO 2

단계 2.3) T의 각각의 output place(P)에 대해서 반복

단계 2.3.1) P가 T의 첫 번째 output place가 아니라면 경로(R1)를 추가. R의 place와 transition의 요소를 상속하고, "New" 로 표시하여 현재 경로는 R1로 한다.

단계 2.3.2) P를 현재 경로에 추가.

단계 2.3.3) P의 output transition이 없거나 R1 의 P가 존재조건 1에 만족하지 않으면 R1을 "End" 로 설정. GO TO 2.3

단계 2.4) P의 각각의 output transition (T1)에 대해서 반복

단계 2.4.1) T1이 P의 첫 번째 output transition이 아니라면 경로(R2)를 추가.

단계 2.4.2) 현재경로의 place와 transition의 요소를 상속. "New" 로 표시하여 현재 경로는 R2로 한다.

단계 2.4.3) T1을 현재경로에 추가.

단계 2.4.4) T1이 존재조건 1을 만족하지 않으면 R2를 "End" 로 설정. GO TO 2.4

단계 2.4.5) T1이 Check Transition 이고,

단계 2.4.6) 현재경로가 존재 조건 1·2를 만족하면, "Find" 로 설정. 그렇지 않으면, 현재 경로를 "End" 로 설정. GO TO 2.4

단계 3) "Find" 로 설정된 각각의 경로의 끝에 있는 check-transition의 block상황을 저장

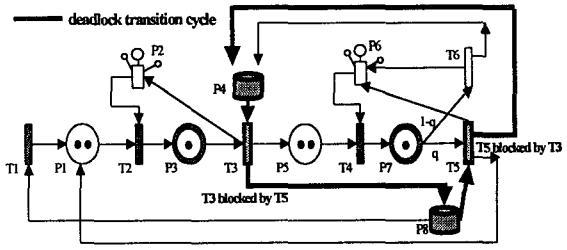
3.4 교착탐지 및 회복

교착이란 하나의 프로세스 s_j에서 대기 중인 고객이 자원이 충족 될 때까지 대기장소에서 대기하는 상태가 정의(blocked)되고, 이 자원은 다른 프로세스 s_i의 보유 자원 중 하나 이상의 자원의 이동에 의해서 충족되며($s_i \rightarrow s_j$), s_i 또한 대기장소에서 대기하는 상태가 정의되며, 이러한 대기 상태의 연속으로 $s_i \rightarrow s_j \rightarrow \ldots \rightarrow s_i$의 하나의 cycle이 정의되는 상황이다[6,12]. 이를 해석하면, $ct(q_i, l) \rightarrow ct(q_j, m) \rightarrow \ldots \rightarrow ct(q_i, l)$의 transition cycle이 정의되는 상황이다. 이러한 사이클이 발생할 경우 전체 system이 정지하는 원인을 제공

하며, 결국 시스템 설계자는 시스템의 올바른 시스템 성능 통계치를 얻을 수 없다. 따라서 이러한 cycle이 모의실험 진행 도중 발생할 경우, 본 연구에서는 존재조건 1, 2와 정리 1을 이용하여 blocked상황과 blocking상황을 정의할 수 있으며, deadlock transition-cycle을 정의할 수 있다. deadlock transition-cycle 발생 시 "Jun and Perros"논문의 가정[4,6]를 응용하여, 일정시간이 경과한 후 cycle 상의 모든 transition을 동시에 발사시킨다. 따라서 시뮬레이션 사건을 계속 발생시킬 수 있으며 봉쇄된 transition의 봉쇄확률, 봉쇄의 원인 및 Deadlock cycle의 구성 transition 등 다양한 결과를 얻을 수 있다.

4. 교착탐지 및 교착회복 예제

4.1 2개의 노드로 구성된 대기 시스템



<그림 6> 확장된 페트리-네트에 의한 교착탐지

앞에서 설명한 2개의 노드로 구성된 대기 시스템에서는 <그림 6>에서와 같이 교착이 발생할 수 있다. <그림 6>에서 P3의 token은 T5의 발사에 의해 이동될 수 있는 buffer space를 기다리며, P7의 token은 T3의 발사에 의해 이동될 수 있는 buffer space를 기다리는 T3→T5→T3의 deadlock transition-cycle이 발생하여, 전체 시스템이 정지된다. 이러한 경우 앞에서 T3과 T5를 순차적으로 발사시켜 토큰을 이동시켜 주면, 두 개 노드의 서버 모두 서비스 상태가 정의될 수 있다. <표 2>는 봉쇄 탐지 이후의 시스템의 상태전이를 나타낸다.

<표 2> 봉쇄탐지 이후의 시스템 상태전이

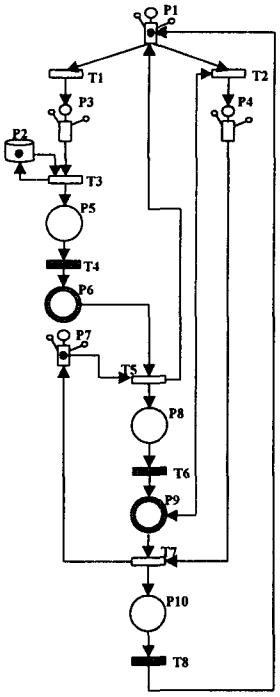
	P1	P2	P3	P4	P5	P6	P7	P8
교착탐지	2	0	1	0	2	0	1	0
T3 발사	2	1	0	-1	3	0	1	1
T5 발사	3	1	0	0	3	1	0	0

4.2 유연생산 시스템

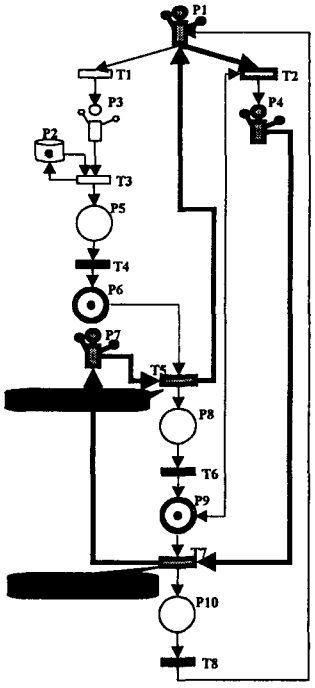
적하장(L/U Station)에는 항상 원자재(Raw Part)가 대기하고 있으며, AGV는 적하장에서부터 NC기계가 있는 장소까지 원자재를 운반한다. 가공이 끝난 부품은 AGV에 의해서 적하장으로 이동된다. AGV는 한번에 하나의 부품만을 운반할 수 있으며, NC기계는 한번에 하나의 부품만을 가공할 수 있다고 가정한다. AGV가 부품을 운반하고, NC기계가 부품을 가공하는데는 일정한 시간이 소요된다. 또한 AGV에 부품이 적재되어 있지 않을 경우 부품을 나르는 시간은 무시할 수 있을 정도로 작다고 가정한다. 초기상태에서 NC기계와 AGV은 유휴상태이다. <그림 7>는 이러한 조건하에서 유연생산시스템을 확장된 페트리-네트 모형으로 표현한 것이며, 각 Place와 transition에 대한 의미는 <표3>과 같다.

<표 3> FMS 시스템의 페트리-네트 기호의 의미

(a) Place
P1 : AGV 대기.
P2 : 사용할 수 있는 원자재.
P3 : 원자재를 운송하기 위해 AGV 대기.
P4 : 운송하기 위해 AGV 대기 완성품 (finished part)위해 AGV 대기.
P5 : AGV는 NC기계로 원자재 이송.
P6 : AGV원자재 이송사태에서 NC기계를 기다림.
P7 : NC기계 .
P8 : NC 기계 가공상태.
P9 : NC기계 가공 후, AGV를 기다림.
P10 : AGV는 완성품 운송
(b) Transition
T1 : AGV 원자재에 할당.
T2 : AGV 완성품에 할당.
T3 : AGV 원자재 운송 시작
T4 : AGV 운송시작.
T5 : AGV 방출. NC 기계 가공시작
T6 : NC 기계 가공.
T7 : AGV 완성품 운송 시작
T8 : AGV 완성품 운송.



<그림 7> 교착이 존재하는 FMS의 확장된 페트리-네트 표현



<그림 8> 교착이 존재하는 FMS의 확장된 페트리-네트의 교착 탐지

모의실험 진행 중에 NC기계가 부품 가공을 끝내고 AGV를 기다리고, AGV은 부품을 적재한 상태에서 NC 기계를 기다리는 상황에서 <그림 8>은 발생한 T5→T2→T7→T5의 deadlock transition-cycle을 나타낸다. <표 4 >은 <그림 6>에서 발생한 transition-cycle 에서 T5, T2, T7의 동시 발사 후 NC기계는 부품을 가공하고, AGV는 적하장으로 부품을 이송하는 상태를 나타낸다.

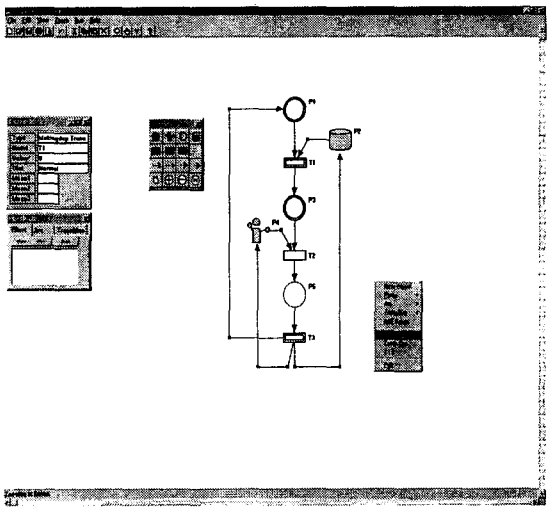
<표 4> 봉쇄탐지이후의 시스템 상태전이

	P1	P4	P6	P7	P8	P9	P10
교착탐지	0	0	1	0	0	1	0
T5 발사	1	0	0	-1	1	1	0
T2 발사	0	1	0	-1	1	1	0
T7 발사	0	0	0	0	1	0	1

5. 실험 및 평가

개발된 소프트웨어의 상용편의성 및 정확도를 입증하기 위하여 많은 실험을 수행하였다. 본 논문에서는 세 가지의 예제를 소개하고자 하며, 모의실험을 통해 구한 해와 수학적으로 계산된 정확한 해를 비교하여 보았다. 비록 본 논문에서는 세 가지의 실험결과만을 제시하였지만, 본 소프트웨어를 이용해 실험 한 모든 예제에서 실험결과는 유사한 행태를 보였음을 밝힌다.

5.1 M/M/m/K/M 대기행렬의 페트리-네트 모형



<그림 9> M/M/m/K/M 대기행렬의 페트리-네트 모형

M/M/m/K/M 시스템의 M대의 기계가 가동되고 있는 공장으로 비유될 수 있다. 각 기계가 고장날 때까지의 가동시간은 평균이 $1/\lambda$ 인 지수분포를 따른다. 고장이 난 기계는 수리소로 보내진다. 수리소에 존재할 수 있는 최대 고객 수는 K이고 고객이 될 수 있는 고객원의 수는 M임을 알 수 있다. 또한 수리소에서 작업하는 수리공의 수는 m명이 있으며 각각의 수리시간은 평균 $1/\mu$ 인 지수분포를 따른다고 가정하자.

이 시스템을 페트리-네트 모형으로 나타내면 다음과 같이 표현된다. 여기서 place1은 공장에서 가동중인 기계를 나타내며, place2는 수리소의 수용능력을 나타낸다. place3은 현재 수리소에서 대기중인 기계를, place4는 유휴(idle)상태인 수리공을 나타내며, place5는 서비스 상태의 수리공과 기계를 의미한다. 시스템에 존재하는 고객의 수와 수리공의 수가 M과 K로 유한하므로 transition 1과 transition 3은 각각 place1, place5의 토큰 수에 의존하는 marking dependent한 발사율(firing rate)로 발사하게 된다. 따라서 transition 1과 transition 3에는 본 소프트웨어에서 지원하는 marking dependent transition을 사용한다. 이 모형의 페트리-네트 모형은 <그림 9>와 같으며, 서비스 율을 변화하면서 100,000단위 시간 동안의 모의실험을 수행한 결과를 정확한 해와 비교하였다. 그 결과는 <표 5>와 같다. <표 5>에서 $\overline{N}$, T, Pbr은 각각 평균 시스템 고객수, 평균체제시간과 도착하는 고객이 봉쇄될 확률을 나타낸다.

<표 5> M/M/m/K/M 시스템의 통계치

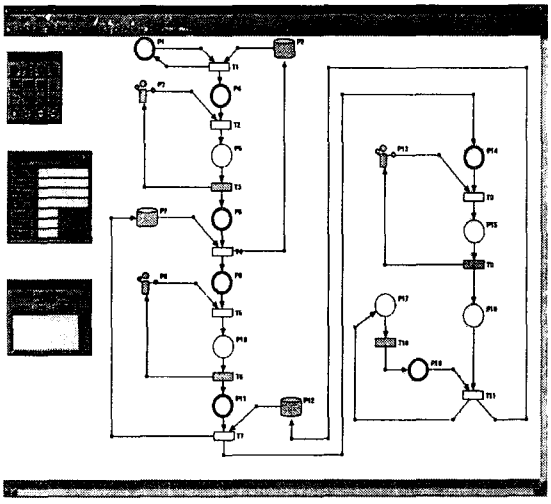
$\lambda=1,\ K=3,\ M=10,\ m=1$				
$\frac{1}{\mu}$	측정치	정확한 해	개발된 S/W	Rel. error (%)
0.2	$\overline{N}$	2.1424	2.1428	0.0186
	T	0.4662	0.4662	0.0000
	Pbr	0.4660	0.4663	0.0643
0.4	$\overline{N}$	2.6121	2.6122	0.0038
	T	1.0610	1.0611	0.0094
	Pbr	0.7037	0.7037	0.0000
0.5	$\overline{N}$	2.7004	2.7006	0.0074
	T	1.3617	1.3624	0.0513
	Pbr	0.7595	0.7597	0.0263
0.7	$\overline{N}$	2.7957	2.7957	0.0000
	T	1.9636	1.9637	0.0050
	Pbr	0.8258	0.8258	0.0000

5.2 간판 시스템(Kanban Systems)

본 절에서는 Mitra[1988]와 Mitrani[1988]의 연구에서 고려되었던 3단계 간판시스템의 예를 분석한다[12].

원자재가 무한하고 3단계의 공정을 거치는 직렬형태 간판시스팀을 고려하자. 완제품에 대한 수요는 비율이 λ인 포아송 과정에 따라 도착하고 모든 가공 공정의 소요시간은 지수분포를 따르며, 목적은 최적 간판 운용 계획을 도출 하고자 함이다. 여기서, 가공 기계는 단계 당 한대씩 있고 시스템 내에 있는 간판의 총 수는6개라고 가정한다. 완제품에 대한 수요의 추후납품(backorder)은 가능하다고 가정하며 각 기계의 가공율은 모두 시간당 3개라 가정한다. 이 시스템을

페트리-네트로 모형화하면 <그림 10>와 같고 <표 6>에는 100,000단위시간 동안의 모의실험 결과치 와 근사적 결과치를 나타낸다. place 2, 7과 12의 토큰 수가 단계 1, 2, 3의 간판의 수로 총 6을 초과할 수 없고, place 3, 8과 13에는 각각 한 개씩의 토큰 즉, 한대의 가공기계가 있다. place 17의 토큰은 제품의 수요를 나타낸다. transition 3, 6과 9는 평균이 1/3인 지수분포를 따르는 가공시간이고 transition 10은 고객이 비율이 λ인 포아송 과정에 따라 도착함을 나타낸다.



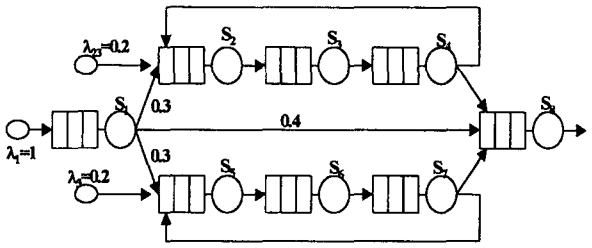
<그림 10> 3단계 간판시스템의 페트리-네트 모형

<표 6> 최적 간판운용 통계치

N_1	N_2	N_3	수리적 해의 산출율(parts/h)	개발된 S/W 산출율 (parts/h)	Rel. error (%)
1	2	3	2.1310	2.1308	-0.0093
3	2	1	2.1310	2.1298	-0.0563
1	3	2	2.2074	2.2082	0.0362
3	1	2	2.1010	2.1016	0.0285
2	1	3	2.1010	2.1002	0.0380
2	3	1	2.2074	2.2092	0.0814
2	2	2	2.1779	2.1805	0.1192
1	1	4	1.9503	1.9491	-0.0615
1	4	1	2.2365*	2.2376*	0.0491
4	1	1	1.9503	1.9514	0.0563

(수리적 해의 산출율 의 Data는 참고문헌[12]의 결과이며, *는 최적 운영계획임)

5.3 봉쇄와 교착이 존재하는 대기행렬 네트워크



<그림 11> 봉쇄와 교착이 존재하는 대기행렬 네트워크

본 절에서는 H.S. LEE, A. Bouhouch, Y. Dallery와 Y. Frein[6]의 분석 대상 예제인 <그림 11>과 같은 봉쇄와 교착이 존재하는 대기행렬 네트워크의 수행도 평가를 위한 모형이다.

본 예제에서의 대기행렬 네트워크는 임의의 형태(arbitrary configuration)로 연결된 네트워크로 8개의 노드로 구성되어 있다. 노드 i(하부 프로세스(S_i))는 각각 유한한 버퍼(finite buffers, B_i)를 가지고 있고, 서어버의 위치를 포함하여 수용도(capacity) C_i를 가지고 있으며, 서비스 시간은 평균 $1/\mu_i$의 지수분포를 따른다고 가정한다. 외부에서 고객의 도착은 <그림 11>에서 보는 것과 같이 프로세스 S_1, S_2, S_3로부터 발생하며, 도착간격은 $1/\lambda_i$의 지수분포를 따른다고 가정하다. 이 시스템에서 blocking mechanism은 전달봉쇄(blocking after service)를 따르고, 한 프로세스에 의해 여러 프로세스가 봉쇄되었을 때 first-blocked-first-enter 원칙을 따른다고 가정한다. 이 시스템은 각각의 프로세스에서 자원이 유한하고 임의의 형태로 연결되어 있어 교착이 발생할 수 있으며, 교착 발생 시 $S_2 \rightarrow S_3 \rightarrow S_4 \rightarrow S_2$과 $S_5 \rightarrow S_6 \rightarrow S_7 \rightarrow S_5$의 deadlock cycle이 발생할 수 있다. 본 예제에서는 Jun and Perros의 논문에서 가정되었던 것처럼 교착이 발생했을 때 즉시 봉쇄된 고객을 동시에

이동시킴으로서 교착을 해소한다고 가정하였다.

H.S. LEE, A. Bouhouch, Y. Dallery와 Y. Frein의 논문에 제시된 시뮬레이션 결과와 본 연구에서 개발된 소프트웨어를 이용한 시뮬레이션 경과는 <표 7>에 비교되어 있다. N_i 과 Pbr_i 은 프로세스 S_i에서의 평균 고객 수와 봉쇄 확률을 의미하며, 종료조건은 100,000시간으로 <표 7>에서 볼 수 있듯이 시뮬레이션의 두 결과는 매우 유사한 것으로 나타났다.

<표 7> 봉쇄와 교착이 존재하는 대기행렬 네트워크의 통계치

	C=(2,2,2,2,2,2,2,2) μ=(1.5,1.0,1.0,1.0,1.0, 1.0,1.0,1.5)			C=(2,2,2,2,2,2,2,2) μ=(2.0,1.5,1.5,1.5,1.5, 1.5,1.5,2.0)		
측정치	H.S LEE	개발된 S/W	Rel. error (%)	H.S LEE	개발된 S/W	Rel. error (%)
발생한 교착수	1296	1283		418	432	
N_1	1.3255	1.3267	0.0904	0.8952	0.8953	0.0112
Pbr_1	0.3068	0.3062	-0.1960	0.1470	0.1471	0.0680
N_2	0.8699	0.8730	0.3550	0.5320	0.5323	0.0563
Pbr_2	0.1240	0.1248	0.6410	0.0470	0.0474	0.8439
N_3	0.8949	0.8978	0.3230	0.5538	0.5552	0.2521
Pbr_3	0.1412	0.1410	-0.1418	0.0611	0.0621	1.6366
N_4	0.9679	0.9723	0.4525	0.6392	0.6416	0.3740
Pbr_4	0.1816	0.1822	0.3293	0.1126	0.1133	0.6216
N_5	0.8766	0.8746	-0.2287	0.5285	0.5306	0.3958
Pbr_5	0.1257	0.1259	0.0025	0.0467	0.0472	1.2058
N_6	0.9004	0.9018	0.1552	0.5514	0.5540	0.4093
Pbr_6	0.1427	0.1435	0.5575	0.0611	0.0617	0.9724
N_7	0.9731	0.9745	0.1437	0.6370	0.6401	0.4842
Pbr_7	0.1830	0.1830	0.0000	0.1122	0.1133	0.9708
N_8	1.2108	1.2125	0.1402	1.0120	1.0124	0.0395
Pbr_8	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

6. 결론 및 향후 연구

본 연구에서는 페트리-네트를 확장하고, 다양한 기능을 부여하여 윈도우(window) 환경 하에서 사용 가능한 모의실험 소프트웨어를 Visual Basic과 Visual C++을 이용하여 개발하였다. 본 연구에서 개발된 소프트웨어는 그래픽도구를 이용하여 사용자가 분석하고자 하는 시스템을 페트리-네트로 표현하여주기만 하면, 모의실험을 자동으로 수행하여 원하는 시스템의 성능척도를 제공하여 준다. 본 연구에서는 봉쇄의 새로운 정의를 통하여 기존 소프트웨어에서는 볼 수 없었던 교착탐지와 교착해소기능을 부여하였다.

본 연구에서 개발된 소프트웨어에 의해 시뮬레이션이 정확히 수행되는지를 입증하기 위하여 많은 예제에 대하여 실험하여 보았다. 그 결과 본 연구에서 개발한 소프트웨어는 신뢰할 만한 시뮬레이션 결과를 제공함을 입증할 수 있었다.

페트리-네트에 대한 기본지식은 쉽게 습득될 수 있으므로 많은 시스템을 페트리-네트로 쉽게 모형화 할 수 있다. 따라서 비전문가라도 본 연구에서 개발된 소프트웨어를 이용하여 대상시스템의 성능 분석을 쉽게 수행 할 수 있으며, 일반적으로 페트리-네트로 쉽게 모형화 할 수 있는 시스템은 많으므로 본 연구에서 개발된 소프트웨어의 실용적 가치는 클 것으로 기대된다.

본 연구에서 개발된 소프트웨어의 완성도를 높이기 위해서 보완 혹은 추가될 점이 많이 있다. 시스템의 논리적 특성 분석을 위해서 reachability tree, animation 등의 기능을 포함한 모듈(module)을 앞으로 추가해야 할 것이며, 사용자 편의성측면에서도 보다 세심한 배려를 해야 할 것으로 판단된다. 만일 페트리-네트의 상태변환과정의 시간 분포가 지수분포를 따르고 시스템의 상태공간이 크지 않을 경우에는 페트리-네트에 내재되어 있는 연속시간 마코프-체인(Continuous Time Markov Chain)을 분석함으로써 시스템의 수리적 성능분석을 수행할 수 있다. 따라서 이러한 수리적 분석기능이 소프트웨어에 추가될 수 있으면 굳이 값비싼 모의실험을 수행

하지 않아도 시스템의 정확한 성능척도를 얻을 수 있으므로 매우 효과적이라 할 수 있으며, 이러한 기능도 본 소프트웨어에 추가되어져야 하리라 판단된다. 추후 저자들은 수리적 성능분석과 모의실험의 장점을 동시에 충족시키며, reachability tree 및 animation 등의 기능을 포함한 통합 소프트웨어의 개발을 완료한 후 이에 대한 내용을 별도의 논문을 통하여 상세히 소개할 예정이다.

참고 문헌

[1] Cassandras, C. G., *Discrete Event Systems*, Irwin, Boston, 1993.

[2] Chiola, G., "A Graphical Petrinet Tool for Performance Analysis," Workshop on Modeling Techniques and Performance Evaluation, Paris, France, 1987, pp 297- 308.

[3] Ciardo, G. and J. K. Muppala, *Manual for the SPNP Package Version 3.1*, Duke Univ., Durham, 1991.

[4] Jun, K. P. and H. G. Perros, "Approximate analysis of arbitrary configuration of queueing networks with blocking, Proc. 1st International Workshop on Queueing Networks with Blocks, North-Holland, amsterdam, 1989, pp 259-280.

[5] Law, A. M., and W. D. Kelton, *Simulation Modeling and Anaysis*, 2nd ed., McGraw -Hill, New-York, 1991.

[6] Lee, H. S., A. Bouhchouch, Y. Dallery and Y. Frein, "Performance Evaluation of Open Queueing Networks with Arbitrary Configuration and Finite Buffers," Annals of Operations Research, 1998, pp 181-206.

[7] Martinez, J. C., *STROBOSCOPE Userguide*, 1998.

[8] Menasche, M., *PetriSim Manual*, 1995.

[9] *Microsoft Visual Basic 5.0 Manual*, Microsoft, 1997.

[10] *Microsoft Visual C++ 5.0 Manual*, Microsoft, 1997.

[11] Saxony, D. *Visual SimNet Userguide*, 1998.

[12] Viswanadham, N.. and Y. Narahri, *Performance Modeling of Automated Manufacturing Systems*, Prentice Hall, Englewood Cliffs, New Jersey, 1992.

● 저자소개 ●

박찬우
1997 : 경희대학교 산업공학과 학사
1999 : 경희대학교 산업공학과 석사
1999~ 현재 : 경희대학교 산업공학과 박사과정
관심분야: 대기행렬이론, 생산·통신시스템 모델링, 시뮬레이션

황상철
1999 : 경희대학교 산업공학과 학사
1999~ 현재 : 경희대학교 산업공학과 석사과정
관심분야: 대기행렬이론, 생산·통신시스템 모델링, 시뮬레이션

이효성
1978 : 서울대학교 산업공학과 학사
1980 : 한국과학기술원 산업공학과 석사
1988 : University of Michigan 산업공학과 박사
1982 ~ 현재 : 경희대학교 공과대학 산업공학과 교수
관심분야: 대기행렬이론, 생산·통신시스템 모델링, 신뢰성공학, 시뮬레이션