韓國港灣學會 第14卷 第1號, pp. 47~55. 2000.

論 文

# DEVS 형식론을 이용한 컨테이너터미널의 객체지향 시뮬레이션에 관한 연구

성경빈* · 정희균* · 박용욱* · 이철영**

# Object-Oriented Simulation of Container Terminal using a DEVS Formalism

*K. B. Sung · H. G. Jung · Y. U. Park · C. Y. Lee*

**Key Words** : 컨테이너터미널(Container Terminal), 시뮬레이션(Simulation), DEVS 형식론(DEVS formalism), 이산사건 시스템(Discrete Event System), 객체지향 시뮬레이션(Object-Oriented Simulation)

## Abstract

In order to cope with the changes of container terminal situation in these days, many simulation studies for container terminal have been accomplished. But previous simulation studies using simulation language have limitations in model representation and difficulties in modeling of large scaled container terminal system. To make these problems better, this paper addresses an object-oriented simulation of container terminal system using a DEVS formalism. The DEVS (Discrete Event System Specification) formalism, developed by Zeigler, supports specification of discrete event system in a hierarchical and modular manner. The formalism provides a mathematical basis for studying discrete event systems with better understood and sounder semantics. In a step of system modeling, a DEVS formalism aims at the exact system modeling that has a basis of semantics and utilizing the object-oriented manner can flexibly cope with the changes of system environment. In this study a model is developed and verified through the simulation of some alternatives.

* 정회원, 한국해양대학교 대학원
** 정회원, 한국해양대학교 물류시스템공학과 교수

# 1. 서 론

컨테이너 화물의 증대, 선박의 대형화, 고속화 등 컨테이너터미널을 둘러싼 해운 환경의 변화로 컨테이너터미널간의 경쟁이 치열해 지면서 터미널의 생산성을 향상시키기 위해 터미널의 확장, 추가 건설 및 장비와 운영 시스템의 개발이 급속히 이루어지고 있다[1]. 이러한 환경 변화에 대처하기 위한 의사결정 수단으로 컨테이너터미널을 대상으로 한 시뮬레이션 연구가 국내외에서 많이 이루어지고 있으나 현재까지 수행된 대부분의 연구는 시뮬레이션 모델이 그 범위가 특정 하부 시스템에 제한되거나 지나치게 추상적이고 단순화되었기 때문에 적용되는 컨테이너터미널 시스템의 빈번한 환경 변화를 정확히 수용하기에는 많은 어려움이 있다[2]. 뿐만 아니라 기존의 컨테이너터미널을 대상으로 한 모델링과 시뮬레이션 연구들은 주로 GASP, GPSS, SIMAN, SIMSCRIPT, SLAM 등과 같은 시뮬레이션 전용 언어를 사용하여 연구되고 있는데 이러한 시뮬레이션 전용 언어들은 이들 언어에서 제공하는 제한된 블록을 사용하여 시스템을 모델링 해야 하므로 시스템을 수학적으로 기술하는 의미론이 불명확하며 모델의 표현력도 제한된다[3]. 이러한 의미론이 불명확함은 컨테이너터미널 시스템의 모델링 시 모델 내부 혹은 외부에서 발생하는 사건의 정의가 모호해져 정확한 시스템 모델링에 어려움이 있다[3,4]. 또한 이들 언어들은 시스템의 각 구성요소들을 모듈화하고 계층적으로 분해하거나 결합시키는 기능이 없으므로 복잡한 시스템의 모델링 작업에 비효율적일 뿐 아니라 컨테이너터미널의 하역 시스템 및 운영 방식, 시설물의 배치 등에서 빈번한 변화를 갖는 현대 컨테이너터미널을 시뮬레이션 하는데 있어 유연성이 떨어지고 모델링 과정의 오류를 검증하거나 모델의 유지보수에 많은 어려움이 있다.

이와 같은 시뮬레이션 언어를 사용한 기존 연구의 단점을 보완하고 컨테이너터미널 시스템의 환경 변화에 유연하게 대처하기 위해서는 객체지향을 바탕으로 한 시스템 설계와 수학적 형식론의 의미론에 기반을 둔 새로운 컨테이너터미널의 모델링 및 시뮬레이션 방법이 요구된다.

본 연구에서는 컨테이너터미널을 대상으로 한 모델링과 시뮬레이션 시 대두되는 모델의 체계적인 분석과 모델링 방법, 그리고 터미널 내에서 빈번히 발생하는 변화에 적절히 대처하기 위한 방법으로 DEVS(Discrete Event System Specification) 형식론과 객체지향 방법론을 이용한 컨테이너터미널의 모델링과 시뮬레이션 방법을 제시한다.

# 2. 접근 방법

## 2.1 DEVS 형식론을 이용한 수학적 모델링

일반적으로 시스템을 수학적으로 모델링 하고자 할 경우 연속 시스템의 모델링에 사용되는 미분방정식과 같은 수학적인 형식론이 사용된다. 그러나 컨테이너터미널과 같은 이산사건 시스템 모델링의 경우 연속 시스템의 모델링에 사용되는 미분 방정식과 같은 일반적인 형식론이 개발된 것이 없다[3,7].

Zeigler에 의해서 제안된 DEVS 형식론[4,5]은 이산사건 시스템을 계층적 구조를 가진 모듈화 된 모델로 명세하는 수학적 형식론으로 모델링 대상인 객체들을 명료하고 객관적인 방법으로 기술함으로써 일관된 시스템의 표현을 가능하게 한다. 형식론을 채택하여 시스템을 관찰하고자 하는 초점에서 추상화하여 모델링 함으로써 모델과 실제 문제와의 관련성을 높일 수 있다[3]. DEVS 형식론은 집합론에 기반을 두고 있으며, 이산사건 시스템을 계층적으로 분해하여 나타내기 위하여 두 가지 모델 클래스, 즉 Atomic DEVS 모델과 Coupled DEVS 모델로 나누어서 시스템을 표현한다.

### 2.1.1 Atomic DEVS 모델

Atomic DEVS 모델은 계층적인 구조로 구성되는 시스템 엔터티 구조에서 더 이상 분해 할 수 없

는 시스템 구성원을 표현하는 모델로 외부사건 또는 내부사건에 의해 상태가 전이하는 동적 모델에 해당하며, 출력은 내부 시간 초과에 의해 상태가 전이 될 때 발생한다[4,5,6]. 이러한 출력은 다른 모델의 외부 사건을 발생시켜 다른 모델의 상태전이를 발생시킨다.

Atomic 모델은 다음과 같이 표현된다.

$AM=\langle X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta\rangle$

여기서,

$X$ : 입력사건 집합, 유한집합

$S$ : 상태변수 집합, 유한집합

$Y$ : 출력사건 집합, 유한집합

$\delta_{ext}$ : $Q\times X\rightarrow S$ : 외부전이 함수

$\delta_{int}$ : $S\rightarrow S$ : 내부전이 함수

$\lambda$ : $S\rightarrow Y$ : 출력 함수

$ta$ : $S\rightarrow Real_0^{+\infty}$ : 시간전진 함수

여기서 $Real_0^{+\infty}$은 0을 포함한 양의 실수의 집합이고 $Q$는 Atomic 모델 $AM$의 전체상태로서 다음과 같이 정의된다.

$$Q=\{(s,e) \mid s\in S \text{ and } 0\le e\le ta(s)\}$$

2.1.2 Coupled DEVS 모델

Coupled 모델은 복합형 구성요소로서, 인터페이스는 Atomic 모델과 같지만 Atomic 모델 또는 다른Coupled 모델의 집합으로 구성되어 다른 더 큰 규모를 가진 모델의 구성요소 모델이 될 수 있기 때문에 이것을 이용하여 복잡한 모델을 계층적으로 구성할 수 있게 된다[5,6].

Coupled 모델은 다음과 같이 표현된다.

$CM=\langle X, Y, M, EIC, EOC, IC, SELECT\rangle$

여기서,

$X$ : 입력사건 집합 , 유한집합

$Y$ : 출력사건 집합 , 유한집합

$M$ : DEVS 구성요소 집합, 유한집합

$EIC \subseteq CM.IN \times M.IN$ : 외부입력 연결관계

$EOC \subseteq M.OUT \times CM.OUT$ :외부출력연결관계

$IC \subseteq M.OUT \times M.IN$ : 내부 연결관계

$SELECT$ : subset of $M \rightarrow M$: 여러 구성요소모델들이 같은 시각에 스케줄을 원할 때 그 중에서 하나를 고르는 함수

여기서 $SELECT$ 함수는 같은 시간에 내부 상태 전이를 해야 할 모델이 여럿일 경우에 그것들을 순서적으로 처리하는 역할을 한다.

2.1.3 추상화 시뮬레이터

추상화 시뮬레이터는 DEVS 형식론에 의해 명세된 모델의 동적인 특성을 해석하여 시뮬레이션 하는 알고리즘으로서, Atomic 모델을 위한 시뮬레이터와 Coupled 모델을 위한 코디네이터의 두 종류로 나뉜다[4]. DEVS 시뮬레이션은 형식론에서 정의된 여러 함수들을 수행함으로써 진행되는데, 이들은 DEVS 추상화 시뮬레이터에서 내부사건과 외부사건을 처리할 때 호출된다. DEVS 추상화 시뮬레이터에는 $\langle *, t\rangle$, $\langle x, t\rangle$, $\langle y, t\rangle$, $\langle done, t_N\rangle$의 네 종류의 메시지가 존재한다. 여기서 $\langle *, t\rangle$와 $\langle x, t\rangle$는 각각 내부사건과 외부사건을 나타내며, $\langle y, t\rangle$와 $\langle done, t_N\rangle$은 출력과 다음 내부 상태 전이를 위한 스케줄 정보의 전달을 위하여 쓰인다.

## 2.2 객체지향 설계

지속적으로 변화하는 컨테이너터미널 환경에 적절히 대응하고 크고 복잡한 컨테이너터미널 시스템을 정확하게 모델링, 시뮬레이션 하기 위해서는 객체지향 방법론을 이용한 대상 모델의 분석과 설계가 요구된다. 특히 DEVS 형식론은 컨테이너터미널과 같은 크고 복잡한 시스템을 계층적으로 모듈화하여 기술할 수 있는 특성을 가지고 있어 컨테이너터미널의 모델링과 시뮬레이션에 매우 적합하다. 컨테이너터미널을 대상으로 한 시뮬레이션의

경우 시스템을 구성하는 각각의 단위 모델들이 객체에 대응하며 단위 모델 사이의 사건 전달은 객체 사이의 메시지 전달로 구현된다. 본 논문에서는 컨테이너터미널을 구성하는 컨테이너, 하역 · 이송 · 장치 장비, 선박, 장치장, 게이트 등 실제 시스템내의 객체들을 각 단위 모델 클래스로 구성, 모듈화하여 복잡하고 변화가 많은 현대의 컨테이너터미널 시스템에 적절히 대응할 수 있도록 구성하였다.

## 3. DEVS형식론을 이용한 컨테이너 터미널의 설계 및 모델링

컨테이너터미널 시스템을 DEVS 형식론으로 모델링 하기 위해 크게 컨테이너터미널 내에서 서비스를 받는 컨테이너 모델과 이를 처리하기 위한 하역 · 이송 · 장치 장비와 장치장, 게이트 등의 자원 모델, 그리고 이들을 통제 관리하기 위한 운영 컨트롤 모델 등의 클래스로 구성하여 모델링 하였다.

본 연구에서 모델링 되어진 컨테이너터미널의 시뮬레이션 모형은 Fig. 1과 같다.

시뮬레이션 모형은 Fig. 1에서 보는 것과 같이 선박, 컨테이너, 트럭을 발생시키기 위한 각각의 Generator Atomic 모델 (Fig. 1에서 Ship_Gen, Con_Gen, Tru_Gen), 결과 통계값을 받아들이기 위한 Transducer Atomic 모델, 그리고 각 자원들의 Atomic 모델, 이들이 결합된 Coupled 모델, 그리고 전체적인 운영을 통제하는 운영 컨트롤로서의 Computer Atomic 모델로 구성되어 있다.

그리고 각 DEVS 모델들이 결합되어 대상 컨테이너터미널의 최상위 계층의 DEVS 모델이 구성된다.

그림에서 굵은 화살표는 컨테이너터미널 내에서의 컨테이너 흐름을 나타내고, 얇은 화살표는 메시지의 흐름을 나타낸다. Atomic 모델 내부에는 상태변수를 나타내는 메모리 변수가 있고 Coupled 모델 내부에는 결합되어진 각각의 Atomic 모델들이

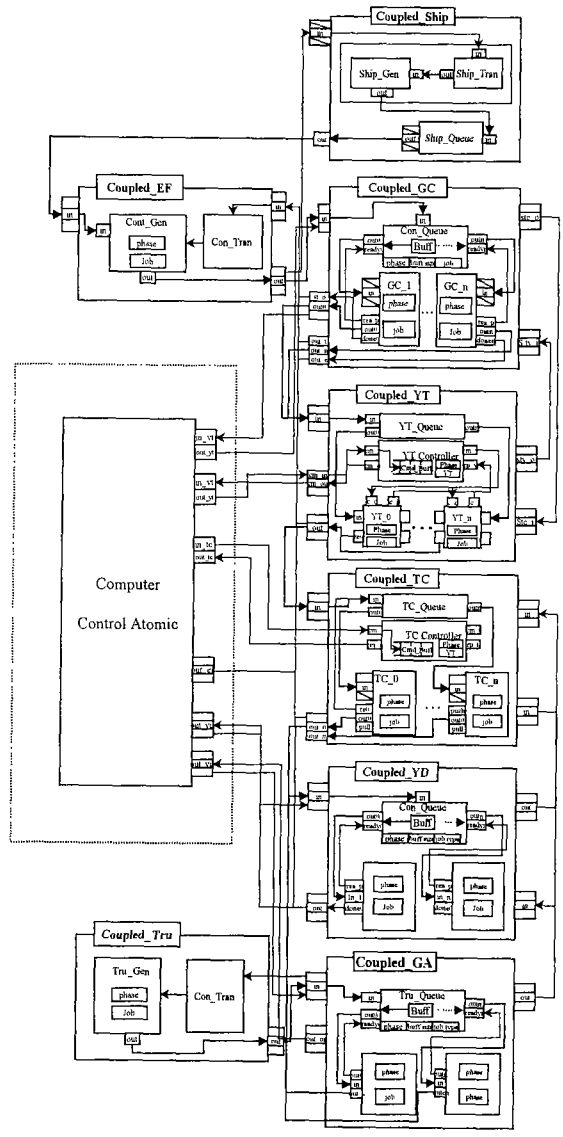


Fig. 1 DEVS Model of Container Terminal System

존재한다. 각 화살표 시작은 출력 단자를 화살표의 끝은 입력 단자를 나타낸다.

### 3.1 컨테이너 모델의 모델링

컨테이너터미널 내에서 처리되는 컨테이너는 자

신이 시스템 내에서 받아야 할 하나 이상의 프로세스 정보를 가지고 있으며 내부 데이터 필드로 Container_Id, Container_Type, Process_Time 등을 가지고 있다. 또 시스템 내에서 컨테이너에 대한 처리가 진행됨에 따라 처리하여야 할 작업이 바뀌게 된다. 컨테이너 발생기에서 컨테이너를 발생시킬 때 컨테이너의 해당 속성이 정의된다. 즉, 컨테이너 발생과 동시에 Container_Type, Container_Id 등을 정의할 수 있다. 컨테이너 흐름 모델링은 YT(Yard Truck)나 GC(Gantry Crane), TC(Transfer Crane) 장치장, 게이트 등의 자원들 사이에서의 컨테이너 이동으로, 해당 자원의 자체 오퍼레이션에 의해 이동이 자율적으로 수행된다.

### 3.2 자원 모델의 모델링

자원 모델은 컨테이너터미널 내에서 컨테이너의 하역, 이송, 장치등 컨테이너를 취급하는 모든 장비와 장치장, 게이트 등이 해당되며, 컨테이너의 위치나 속성을 변화시키는 능동적 자원과 속성이나 위치의 변화 없이 단순히 저장, 대기하는 수동적 자원으로 구분한다. 또한 이들 자원은 이동 가능한 자원과 이동 불가능한 자원으로 나눌 수 있다.

이동 가능한 자원은 자신의 위치를 변경시킬 수 있고 생성하고 소멸될 수 있으며 이동 불가능한 자원은 자신의 위치를 변경할 수 없는 자원을 말한다. 이동 가능한 자원에는 GC, TC, YT등과 같이 자체 구동 능력을 가지는 자원을 말하며, 이동 불가능한 자원은 장치장, 게이트 그리고 하역, 이송을 담당하는 자원들이 작업을 대기하는 대기열 등을 말한다. 각 자원에 대한 모델의 명세화 내용은 상태전이 다이어그램으로 표현한다. 상태는 타원 내에 표시되며 상태변수인 메모리 변수는 사각형 내부에 표시한다.

본 논문에서는 DEVS 형식론을 이용한 컨테이너 터미널 모델링 방법의 한 예로 자원들을 나타내는 많은 Atomic 모델, 혹은 Coupled 모델들 중 전형적인 모델로서 Coupled_GC(Gantry Crane) 모델의 예를 들어 모델링 수행 절차를 설명한다.

모델링 단계로는 먼저 해당 자원에서 서비스를 받는 컨테이너를 모델링하고 DEVS 형식론에 따라 모든 자원을 구성하는 Atomic 모델에 대해 $S$, $X$, $Y$, $ta$ 를 정의하고 메시지 교환을 위한 메모리 변수도 함께 정의한다. 운영 컨트롤의 모델링 단계로 각각의 자원과 자원에서 처리해야 할 컨테이너의 짝을 맺어주고 $\delta_{ext}$, $\delta_{int}$, $\lambda$ 를 정의한다. 이렇게 구성한 각각의 Atomic 모델들을 결합하여 상호 결합 관계를 정의함으로써 모델이 완성된다.

Coupled_GC 모델은 Fig. 1에서 보는 것과 같이 한 개의 Con_Queue Atomic 모델과 N개의 GC Atomic 모델로 구성되어 있다.

먼저 Cont_Gen Atomic 모델에서 발생된 컨테이너가 Coupled_GC의 입력단자인 "in"으로 들어오면 Con_Queue Atomic 모델에서 내부 버퍼에 저장 한 다음 Container_type과 GC 상태를 검사하여 해당되는 GC에 컨테이너를 배정하면 GC는 Container_type에 따라 정해진 시간의 작업을 수행하여 Coupled_GC 모델의 출력단자 out1~out $n$을 통해 컨테이너를 보내고 컨테이너 처리·대기 정보를 Con_Queue에 보낸다.

GC를 나타내는 GC_n Atomic 모델은 입력단자 "in"으로부터 컨테이너를 받으면 컨테이너를 상태변수 Cont에 저장한다. 컨테이너 처리시간이 끝나면 Phase를 Busy에서 Block으로 바꾸고 출력단자 "Out"을 통해서 Computer Atomic 모델에게 작업완료를 알린다. 그리고 입력단자 "St_in"으로부터 Send 정보를 받으면 출력단자 "Out"을 통해서 YT로 컨테이너를 보내고 Phase가 Idle이 된다.

Con_Queue Atomic 모델과 GC Atomic 모델의 명세화 내용을 상태전이 다이어그램으로 표현하면 Fig. 2, Fig. 3과 같다.

Fig. 2는 Con_Queue Atomic 모델의 상태전이 다이어그램을 나타낸 것이다.

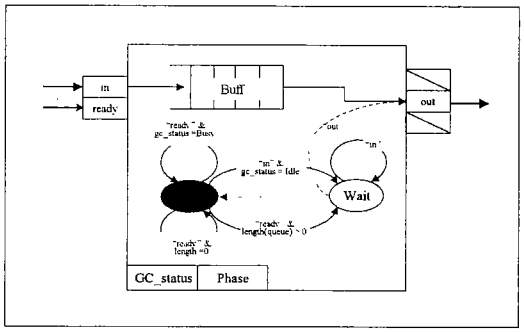


Fig. 2 Phase Transition Diagram (PTD) of Con_Queue Atomic Model

Fig. 3은 GC_n Atomic 모델의 상태전이 다이어그램을 나타낸 것이다.

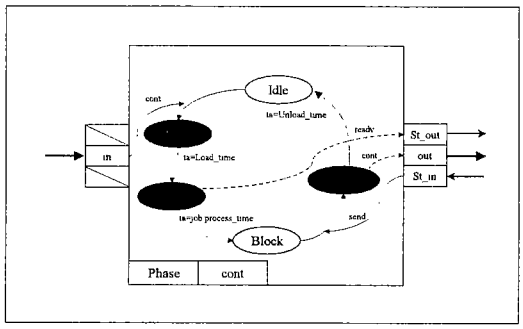


Fig. 3 Phase Transition Diagram (PTD) of GC_n Atomic model

Con_Queue Atomic 모델은 다음의 Table 1과 같이 DEVS 형식론으로 나타낼 수 있다.

GC_n Atomic 모델은 Table 2와 같이 DEVS 형식론으로 나타낼 수 있다.

Coupled 모델에 의해 표현되는 YT, TC는 한 개의 YT, TC Controller와 N개의 YT, TC Atomic 모델로 구성되어 있다. 이송 · 장치 장비에 해당하는 YT와 TC의 모델링을 위한 상태전이 다이어그램은 Fig. 4 - Fig. 7과 같다.

Table 1 Pseudo Code of Con_Queue Atomic model

$AM=\langle X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta\rangle$

| | |
|---|---|
| $X$ | $X$ = {in, ready} |
| $Y$ | $Y$ = {out} |
| $S$ | $S$ = {GC_status, phase}<br>phase = {Send, Wait} |
| $\delta_{ext}$ | $\delta_{ext}$ : case input event of "in"<br>insert(x, queue);<br>if (phase == Wait and<br>GC_status == Idle)<br>then phase :== Send;<br>else continue;<br>case input event of "ready"<br>GC_status := Idle;<br>if (!empty(queue))<br>then phase := Send;<br>else continue; |
| $\delta_{int}$ | $\delta_{int}$ : if (phase == Send)<br>then delete(first queue);<br>GC_status := Busy;<br>Phase := Wait; |
| $\lambda$ | $\lambda$ : if (phase == Send)<br>then ("out", "first(queue)"); |
| $ta$ | $ta$ : if (phase == Send)<br>then ta := Sending_time(first(queue);<br>if (phase == Wait)<br>then ta := infinity; |

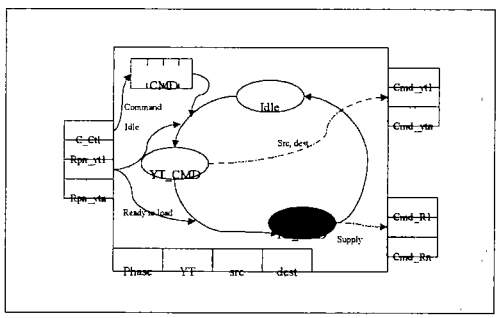


Fig. 4 PTD of YT Cotroller Atomic Model

Table 2 Pseudo Code of GC_n Atomic model

$AM=\langle X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta\rangle$

| | |
|---|---|
| $X$ | $X$ = {in, send} |
| $Y$ | $Y$ = {ready, out} |
| $S$ | $S$ = {phase, job}<br>phase={Idle, Load, Busy, Block, Send} |
| $\delta_{ext}$ | $\delta_{ext}$ : case input event of "in"<br>insert(cont, job);<br>if (phase == Idle)<br>then phase := Load;<br>else continue;<br>case input event of "ready"<br>if (phase == Block )<br>then phase := Send;<br>else continue; |
| $\delta_{int}$ | $\delta_{int}$ : if (phase == Load)<br>then phase := Busy;<br>if (phase == Busy)<br>then phase := Block; |
| $\lambda$ | $\lambda$ : if (phase == Busy)<br>then ("St_out", "ready");<br>if (phase == Send)<br>then ("out", "job"); |
| $ta$ | $ta$ : if (phase == Load)<br>then ta := Load_time;<br>if (phase == Busy)<br>then ta := job.process_time;<br>if (job == Send)<br>then ta := Unload_time; |

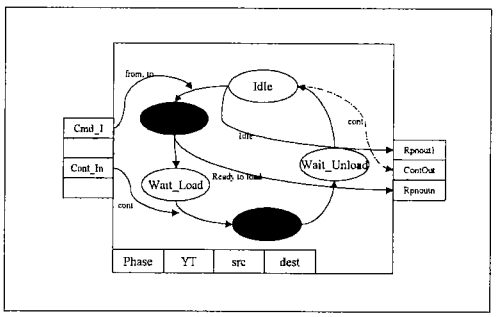


Fig. 5 PTD of YT_n Atomic Model

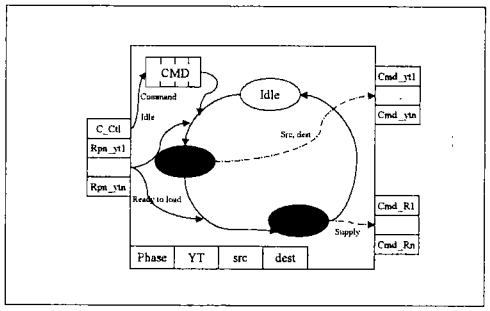


Fig. 6 PTD of TC_Controller Atomic Model

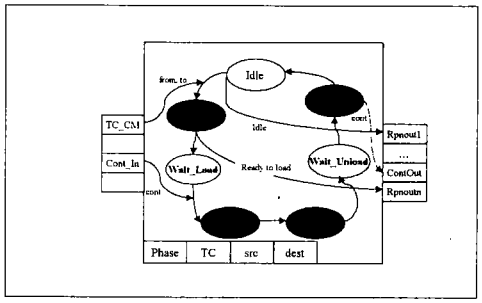


Fig. 7 PTD of TC_n Atomic Model

이러한 방법으로 각각의 자원에 대해 DEVS 형식론에 맞게 정의한 다음 모델을 연결하여 컨테이너터미널의 전체 모형을 구성할 수 있다.

## 4. 시뮬레이션 모형의 실험

본 논문에서는 모델링 되어진 컨테이너터미널 모델을 실험하기 위하여 기존의 관련 연구와 비교하여 시뮬레이션을 수행하고 모델을 검증하였다

비교되어진 연구는 1998년 자성대 부두를 대상으로 IMF에 의한 컨테이너 물류시스템의 영향을 파악해 보기 위해 연구되었다[8].

여기에서는 시뮬레이션을 위해 AweSim 시뮬레이션 언어를 사용하였으며 1998년 이후 6개월 동안의 자성대 부두의 실제 전산 데이터를 사용하여 시뮬레이션을 수행하였다. 제안된 컨테이너터미널의

DEVS모델을 검증하기 위해 두 가지 대안을 비교하여 시뮬레이션을 수행하였다.

첫째, 비교 대상 모델과의 정확한 비교를 위하여 다음과 같은 사항을 가정하여 모델링 하였다. 실험 대상 터미널인 자성대 부두의 경우 장치장에서의 부분적인 혼합 시스템을 사용하고 있다. 그러나 비교 연구에서는 장치장의 컨테이너 장치 장비를 고려하지 않고 장치장의 장치 능력만을 고려하였다. 실험모델에서도 이와 같이 장치장의 장치 장비를 고려하지 않고 장치장 모델의 저장 능력만을 고려하여 모델링을 수행하였다.

둘째, 장치장에서의 컨테이너 장치에 사용되는 장치 자원들을 고려하여 시뮬레이션을 수행하였다. 이는 기존의 장치 자원을 고려하지 않고 장치능력만을 고려한 모델과의 비교뿐만 아니라 본 논문에서 제안한 컨테이너 터미널의 환경변화에 대한 모델링의 적용 용이성을 보이기 위해 실험하였다. 장치장에서의 컨테이너의 하역은 TC에 의해서만 수행된다고 가정하였다.

시뮬레이션을 위한 매개변수 추정치들은 다음과 같다.

- 선박의 평균도착 시간 간격 (시간) : Expon(6.17)
- 하역 작업 준비시간 : 1.83 시간
- 하역 컨테이너 수(TEU) : Normal(653, 269), Expon(1802)를 이용한 수정 정규분포
- GC의 하역 능력 (hr/TEU) : 0.04
- TC의 상하차 능력 (hr/TEU) : 0.04
- TC의 수 : 25기
- YT의 이송 능력 (hr/TEU) : Uniform(0.17, 0.2)
- YT의 수 : 56대
- 장치지역의 장치능력 (TEU/일) : 12,156
- Gate의 작업능력 (hr/TEU) : 0.04(in), 0.02(out)
- Gate의 Lane 수 : 5(in), 3(out)

본 연구에서 실험 모델의 시뮬레이션은 DEVS 형식론을 C++언어로 표현한 DEVSim++을 이용하였다. Warming-Up 시간은 고려하지 않고 시뮬레이션 시간을 240시간으로 하여 시뮬레이션을 실시하였다.

시뮬레이션 결과는 Table 3과 같다.

Table 3 Simulation Result of Experiment model

| 시뮬레이션 결과 | | | |
|---|---|---|---|
| 구 분 | 기존 연구 | 실험모델 1 | 실험모델 2 |
| 선박평균 대기시간 | 2.95 시간 | 2.88시간 | 2.43시간 |
| GC 이용율 | 49.8% | 50.6% | 54.3% |
| TC 이용율 | - | - | 58.1% |
| YT 이용율 | 57.5% | 58.2% | 63.4% |
| 평균 장치율 | 56.0% | 58.8% | - |

Table 3에서, 실험모델1과 실험모델2 모두 기존의 시뮬레이션 연구와 유사한 결과 값을 얻을 수 있어 제안된 모델의 구성이 적합하다는 것을 확인할 수 있었다. 그리고 모델링 단계에서 대안에 따라 모델을 달리 작성할 필요 없이 기존에 구현되어진 모델에 추가 혹은 변경된 사항들만을 고려 연결관계를 맺어 줌으로서 서로 다른 대안에 따른 모델을 쉽게 구현할 수 있었다.

## 5. 결 론

본 논문에서는 현대의 컨테이너터미널과 같이 시스템이 복잡하고 장비나 운영의 빈번한 변화를 가지는 시스템을 시뮬레이션 하는 방법으로 DEVS 형식론과 객체지향 방법론을 이용한 컨테이너터미널의 시뮬레이션 방법론을 제안하고 기존의 연구와 비교하여 그 적합성을 검증하였다.

DEVS 형식론을 이용하여 대상 시스템을 모델링 함으로써 정확한 시스템의 표현이 가능하며 객체지향 설계를 통해 향후 예상되는 시스템 내의 변화에 유연하게 대처할 수 있음을 알 수 있었다.

그러나 본 논문에서는 시스템의 모델링 방법과 설계 방법에 주안점을 두었기 때문에 실제 컨테이

너 터미널에서 이루어지는 계획문제, 의사결정 문제 등 많은 부분이 고려되지 않아 시뮬레이터로는 아직 부족한 면이 많이 있다.

본 연구의 최종 단계는 DEVS 형식론을 이용한 객체지향 시뮬레이터의 개발에 있으며 입력 자료와 출력 자료 분석을 위한 모듈의 개발, 모델링 단계에서 쉽게 대상 시스템을 모델링 할 수 있는 모델러의 개발, 3차원 그래픽 라이브러리를 이용한 애니메이션 모듈의 개발이 진행되고 있다.

## 참고문헌


1) 이철영, "항만물류시스템", 효성출판사
2) 김우선, " 컨테이너 터미널 운영 개선을 위한 시뮬레이션 시스템 설계", 한국항만학회 추계 학술 대회 논문지, pp. 125-135, 1998.
3) 안명수, 박성봉, 김탁곤 "DEVSim++ 의미론에 기반한 이산사건 시스템의 객체지향 모델링 및 시뮬레이션 환경", 한국정보과학회 논문지 제 21권 제 9호, 1994.
4) Zeigler, B.P., "Object-Oriented Simulation with Hierarchical, Modular Models, Academic Press, 1990.
5) Zeigler, B.P., "Hierarchical Modular Discrete-Event modeling in an Object-Orient environment", Simulation, 1987, pp. 219-230.
6) Tag G. Kim, "The DEVS formalism : hierarchical modular systems specification in C++", Proc. 1992 European Simulation Conference, pp.152-156, 1993.
7) Tag Gon Kim, DEVSim++ User's Manual: C++ Based Simulation with Hierarchical Modular DEVS Models, Computer Engineering Lab., KAIST, 1994.
8) 임봉택, 이재원, 성경빈 "시뮬레이션에 의한 컨테이너터미널 물류시스템의 분석에 관한 연구", 한국항만학회, 1998.