2001년 한국정보처리학회 춘계 학술발표논문집 제8권 제1호

순환공학 환경에서의 실시간 시스템 개발 및 검증을 위한 코드 변환기 설계

고 현*, 조상규, 김광종, 이연식
군산대학교 컴퓨터정보과학과
e-mail : kogo@cs.kunsan.ac.kr

Design of Code Converter for Development and Verification of Real-Time System in Software Round-Trip Engineering Environment

Hyun Ko*, Sangkyu Joe, Kwang Jong Kim, Yonsik Lee
Dept. of Computer Information Science, Kunsan National University

요 약

본 논문은 ATM(Abstract Timed Machine)으로 명세된 실시간 시스템에 대한 재/역공학 측면에서의 개발 및 검증을 위한 코드 변환기를 설계한다. ATM은 모드(*mode*), 전이(*transition*), 포트(*port*)로 구성되는데, 순공학 과정에서 실시간 시스템을 설계, 명세 하는 기존의 정형기법과는 달리 ATM은 소프트웨어의 순환공학 과정에서 사용하기 위해 설계되었다. ATM은 기존 정형기법이 순공학 과정에서의 특정 물리적 환경에서 실행되는 동적행위에 대한 부적절한 표현에 대해 순환공학에서 실시간 시스템의 속성은 물론 특정 환경과 동적 정보 등을 명세하기 위한 정형 기법으로서, 본 논문에서는 DoME을 이용하여 ATM 명세도구를 개발하고 이를 이용하여 실시간 시스템의 특정 요구사항을 위한 ATM을 명세한다. 또한 해당 ATM을 DoME/ATM 스크립트 파일로 저장하고 이에 대한 명세분석을 통해 노드와 관련된 정보를 추출하여 다른 분석도구가 이용할 수 있도록 DB에 저장하거나 매개 언어인 SRL/ATM으로 변환하며, 이러한 SRL/ATM으로 부터 실행코드에 대한 관련 정보를 추출하여 실시간 시스템 개발 및 검증을 위한 Ada 코드를 생성할 수 있는 코드 변환기를 설계한다.

1. 서론

실시간적 발생 상황에 대해 정확한 처리와 예외상황에 대한 신속하고 적절한 조치를 요하는 특성을 가진 임무위급 시스템(*mission-critical real-time system*)은 특정 시간간격 내에 특정 입력에 대한 결과를 발생시켜야 하는 시스템이다. 따라서 특정 상황에 대한 시스템의 여러 동적 행위를 표현하고 다양한 속성을 명세할 수 있는 정형기법이 필요하다. 그러나 실시간 시스템을 명세하는 기존의 정형기법들[1, 2, 5, 7, 8, 9]을 통해 역공학 과정에서 소프트웨어의 동적 행위를 이해하는 것은 매우 어렵다. 이는 기존의 정형기법들이 순공학 과정에서 실시간 시스템을 설계, 명세하기 위해 고안되어 실제 시스템이 특정 물리적 환경에서 실행될 때의 동적 행위를 완전하게 표현하지 못하기 때문이다. ATM은 많은 실시간 시스템의 속성 기술은 물론 순환공학 과정에서 특정환경과 동적정보 등을 명세하기 위해 개발된 정형기법으로 이러한 명세를 분석, 검증할 수도 있다[10]. 본 논문에서는 DoME을 통해 ATM을 명세하고 매개언어인 SRL 코드로의 변환시 ATM 명세를 분석하며, SRL 코드를 Ada 실행코드로 변환하는 코드변환 시스템을 설계한다. 설계된 코드변환 시스템에 의해 생성된 Ada 실행코드는 순환공학 환경에서 재/역공학 과정에 대한 설계 및 검증도구로 사용된다. 본 논문은 2장에서 실시간 시스템을 명세, 분석, 검증할 수 있는 정형기법인 ATM에 대해 설명하고 3장에서는 DoME을 통해 ATM 명세도구를 개발하고 명세도구를 이용, DoME/ATM을 명세하여 스크립트 코드를 생성하며, 4장에서는 DoME/ATM 스크립트 코드를 SRL/ATM으로 변환하는 과정과 SRL/ATM을 Ada 실행코드로 변환하는 코드변환 시스템을 설계한다. 5장에서 결론 및 향후 연구과제에 대해 기술한다.

본 연구는 한국과학재단 특정기초연구(과제번호 1999-2-303-003-3) 지원으로 수행되었음

2. ATM 명세

ATM은 계층성, 분산성, 실시간성, 우선권, 다수에 의한 동기화, 예외처리, 실행성 등 다양한 속성을 가진 실시간 시스템을 명세하기 위해 개발된 LTS (Labeled Transition System)로서 순공학 과정을 위해 개발된 정형기법이 가지는 단점을 해결하고 보다 효율적이고 소프트웨어의 다각도 정보를 제공하기 위한 정형 기법이다.

2.1 ATM 구성 요소

ATM의 각 구성 요소들에 대한 표기는 다음 [표 1]과 같다.

종류		표기	설명
Machine		M:T	T 타입의 M 머신
		(Caller)>M	Caller에서 호출한 M 머신
		(Caller)>M<int, int>	Caller에서 두개의 int형을 매개변수로 인스턴스 되는 generic M 머신
		M:T / A / M:A	추상화 모드에 의한 머신의 계층성을 표현한 것으로 머신 M에 속한 추상화 모드 A의 확장을 표현
Mode	일반모드		실행 가능한 문장을 가진 모듈
	추상화모드		아키텍쳐 구조상의 하위 단계의 머신을 추상화한 모드
	주제모드		각 주제에를 표현하기 위한 개념적 모드
Point	시작점		실행의 시작점
	종료점		실행의 종료점
Port	활성화		머신 활성화
	엔트리		엔트리
	대체		대체
Transition	일반전이		일반적으로 모든 사이에 발생하는 전이
	추상화전이		추상화 모드에 의해서 발생하는 전이

[표 1] ATM의 각 구성 요소 표기

2.1.1 머신

머신은 ATM의 기본 단위로 내부에 자신의 모드를 포함하며, 활성화 시 내부의 흐름을 스스로 제어한다.

2.1.2 모드

계산 모드(computational mode) : 머신 내의 제어와 흐름을 기준으로 전이와 관련 있는 이벤트와 조건을 제외한 변수의 집합, 실행문의 집합을 내부적으로 직접 포함 한다.

추상화 모드(abstract mode) : 소프트웨어의 계층관계를 표현하기 위한 방법으로 구조상 하위 계층에 속하는 기능을 상위 계층에서 추상화 시키기 위한 모드이다.

주제 모드 (subject mode) : 소프트웨어의 특정 기능을 수행하고 있는 상태들을 포함하는 모드로서 예외 처리나 주기적 동작과 같은 다수의 ATM의 주제를 표현한다.

2.1.3 전이 - 모드에서 모드로의 제어의 이동

일반 전이 : 전이이벤트, 조건, 시간 제약을 레이블로 가지며 일반적인 제어의 흐름을 나타낸다.

추상화 전이 : 추상화 모드와 이를 상세화 한 머신 사이의 전이를 나타낸다.

2.1.4 포트 - 머신 사이의 데이터를 송, 수신하는 곳.

활성화 포트(Activation Port) : 머신을 활성화 또는 비활성화 시키는 시그널을 위한 포트

엔트리 포트(Entry Port) : 머신 간 데이터 전송을 위한 포트

대체 포트(Substitution port) : 대체 이벤트 호출을 위한 포트

2.2 ATM 특징

ATM은 추상화 모드를 이용하여 소프트웨어의 아키텍처를 계층적으로 표현한다. 아키텍처의 각 단계의 머신은 추상화 모드를 이용하여 하위 단계 머신의 동작을 포함할 수 있다. 이러한 아키텍쳐 구조는 시스템의 동적 행위를 계층적으로 표현하여 실행 단계에서의 상태와 행위에 대한 예상을 용이하게 하고 각 머신의 호출관계 및 인스턴스 관계를 명시적으로 보여줌으로써 시스템 전체의 실행을 추적할 수 있다.

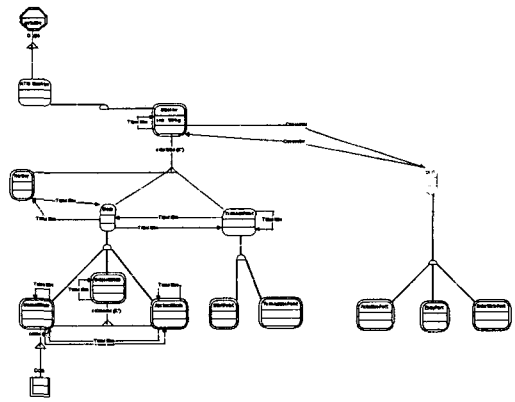
ATM의 모드는 전이에 해당하는 이벤트나 컨디션이 발생하기 전까지의 상태들을 포함하므로 기존의 상태도에서 예상되는 상태폭발(state explosion)을 줄일 수 있다. 모드가 하나 이상의 상태를 포함한다는 것은 시스템 검증 시 검증해야 할 상태가 감소한다는 것을 의미한다.

3. ATM 명세에 대한 DoME 스크립트 코드

3.1 ATM 명세도구

ATM 명세도구는 모델링 정의 도구인 HoneyWell사의 DoME을 이용하여 개발하였다. DoME은 사용자가 필요로 하는 모델링 언어를 개발할 수 있는 환경을 지원하는 메타 모델링(Meta Modeling) 도구이다.

[그림 1]은 DoME을 이용해 개발한 ATM 명세도구의 DoME 명세이다. DoME 명세 파일에서는 ATM에서 필요로 하는 여러 아이콘과 각 아이콘의 상속관계 및 연관관계, 각 아이콘의 모양, 메소드 등과 같은 아이콘이 포함하는 요소들을 명세 한다. 이러한 DoME 명세 파일을 통해 ATM 명세모델로 생성된다.



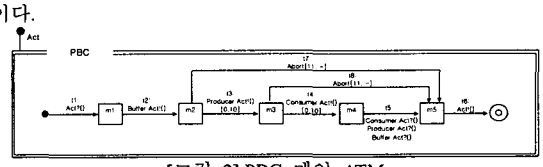
[그림 1] DoME을 이용한 ATM 명세도구 모델

3.2 ATM 명세도구를 이용한 명세 예제

다음은 ATM 명세도구를 이용하여 순환공학 과정에서 PBC (Producer-Buffer-Consumer) ATM을 명세 하는 예제이다. PBC에 대한 다음과 같은 요구사항이 있다고 가정한다:

1) PBC는 Producer, Buffer, Consumer의 독립된 동작으로 구성한다.
2) Producer와 Consumer는 10초 이내에 활성화된다.
3) Producer 문자열을 입력 받아서 5초 이내에 Buffer에 쓴다.
4) Consumer는 Buffer로부터 5초 이내에 문자열을 읽어 출력한다.

다음 [그림 2]는 PBC ATM에서 생성된 PBC 메인 ATM의 명세이다.



[그림 2] PBC 메인 ATM

요구사항으로부터 Producer, Buffer, Consumer 태스크들과 이를 관리하는 PBC 메인 태스크를 추가한 네 개의 머신으로 구

성되는 아키텍처가 결정된다. 이들 머신 간에 존재하는 활성/비활성화 통신과, Buffer의 엔트리 포인트를 통한 통신에 기반하여 시스템을 구성하는 독립된 머신과 그들 사이의 통신관계에 따른 계층구조를 형성한다

3.3 DoME/ATM 명세에 대한 스크립트 코드

ATM 명세도구를 이용하여 특정 시스템을 명세하면, 명세된 그래픽 표현 내용이 각 아이콘의 정보를 기준으로 DoME에서 제공되는 기능에 의해서 자동적으로 텍스트로 저장된다. 다음 [표 2]는 PBC ATM에서 세개의 머신을 관리하는 PBC 메인 ATM의 스크립트 코드이다.

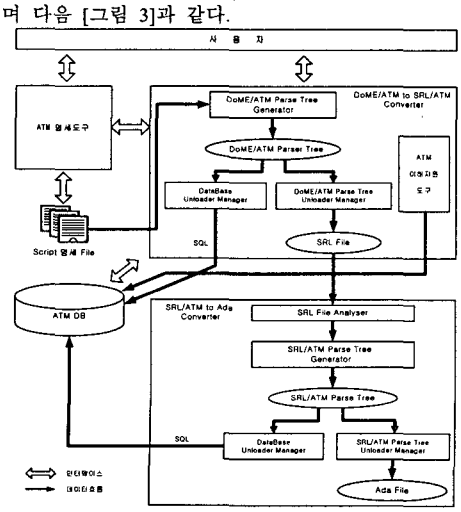
```
# DoME 5.3 model; type: ATM
{ProtoDoMEGraph                    // ATM 명세모델
  modelTypeName: 'ATM'!
  specFile: 'd:\.met'!
  nodes: [OrderedCollection
    [GraphParentLabel
      position: [Point 23210 35]
      id: 2059181063537111'
    ]
    {ProtoDoMENode                          // PBC 메인 머신
      name: 'PBC'!
      position: [Point 440 173]
      size: [Point 727 154]
      components: {PartitionedCollection
                  key: #elements!
                  value: [PartitionedCollection
                    key: #substates!
                    value: [SortedCollection
                      [ProtoDoMENode                    // 상태 m1
                                  :
```

[표 2] PBC 메인 머신에 대한 상태노드 스크립트 코드

4. DoME/ATM의 Ada 실행코드로의 변환

ATM 명세는 그래픽 표현 방식을 사용하고 있기에 코드 변환기 구현 시 자동화 분석을 하기가 매우 어렵다. 따라서 DoME을 통해 생성된 스크립트 코드로부터 DoME/ATM 파스트리 생성기를 이용해 명세 정보를 추출하여 실행코드를 생성하는 과정에서 매개언어인 SRL을 이용하였다. SRL(Software Representation Language)은 시스템을 구성하는 소프트웨어, 하드웨어, 통신망, 형상, 요구사항, 설계, 성능 등에 관한 정보를 저장소에 보관, 관리하고 이를 재/역공학 과정에서 사용자의 특정 목적에 부합하게 표현하여 분석, 이해 및 평가를 하기 위한 매개 언어이다. 이러한 SRL을 생성하여 이를 이용해 Ada, COBOL, C와 같은 다양한 실행코드로의 변환을 수행할 수 있다.

전체 변환 시스템의 구조는 상위부분(DoME/ATM to SRL/ATM Converter)과 하위부분(SRL/ATM to Ada Converter)으로 구성되며 다음 [그림 3]과 같다.



[그림 3] 순공학 과정에서의 코드 변환 시스템 구성도

4.1 DoME/ATM의 SRL/ATM으로의 변환

DoME/ATM 명세파일에서는 ATM에서 필요로 하는 여러 아이콘과 각 아이콘의 상속관계 및 연관관계, 각 아이콘의 모양, 메소드 등과 같은 아이콘이 포함하는 요소들을 명세한다. 이렇게 생성된 ATM 명세도구를 이용하여 시스템을 명세하면 각 아이콘 관련 스크립트 코드는 DoME/ATM 파서를 통해 SRL 코드로 생성된다. [표 3]은 PBC 예제에서 머신 아이콘에 대한 SRL 코드의 표기이다.

종 류	ATM 표기	SRL/ATM 표기
Machine	Producer : Producer_Type	Machine {Name: Producer : Producer_Type Out: {{Name: Type: PortConnector From: Producer : Producer_Type To: Act}}}
	Buffer Read Buffer::Read	Machine {Name: Buffer Out: {{Name: Type: PortConnector From: Buffer To: Act} {Name: Type: PortConnector From: Buffer To: Read} : AbstractMode {Name: Out: {{Name: t6 Type: NormalTransition From: To: m5}}} : Machine {Name: Buffer::Read}

[표 3] Machine 아이콘에 대한 SRL 코드

[그림 3]의 상위부분은 ATM 명세도구에 의해 생성된 스크립트 형식의 ATM 명세를 입력 받아 매개언어인 SRL 파일을 생성하는 변환기에 대한 구성도와 구성 요소들간의 상호작용 과정을 나타낸다.

SRL/ATM 변환기의 동작과정은 먼저 ATM 명세도구를 이용해 생성한 스크립트 형식의 ATM 명세를 DoME/ATM 파스트리 생성기를 통해 ATM 각 구성요소에 관한 정보를 추출하여 매개적인 단계로써 파스트리(DoME/ATM 파스트리)를 구성한다. 이렇게 DoME/ATM 파스트리에 추출된 정보는 데이터베이스 Unloader에 의해 ATM 데이터베이스에 저장되어 ATM 이해 지원도구가 명세에 대한 분석동작을 수행할 수 있도록 기본정보를 제공한다. 또한 DoME/ATM 파스트리에 추출된 명세의 각 구성요소에 대한 계층구조 및 기본정보는 DoME/ATM 파스트리 Unloader를 통해 .s 파일형식의 SRL 파일 형태로 추출되어진다. 구현한 변환기의 구성요소인 파스트리 생성기, 데이터베이스 Unloader, 파스트리 Unloader에 대한 설명은 SRL/ATM에서 Ada 실행코드로 변환하는 변환기 설계부분에서와 기능이 유사하므로 다음 과정에서 설명한다.

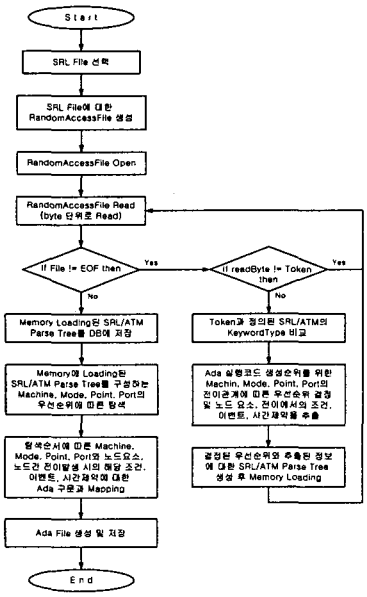
4.2 SRL/ATM의 Ada 실행코드로의 변환

DoME/ATM의 그래픽 표기에서 DoME의 기능을 이용해 변환된 스크립트 코드로부터 검증을 위한 자동화 코드를 생성하기는 어렵다. 따라서 명세에 대한 분석을 통해 생성된 매개언어인 SRL로의 변환으로 자동화 코드의 생성을 비교적 쉽게 할 수 있다. 이는 객체지향 언어나 구조적 언어의 모듈구조에 대한 기본 정보를 제공함과 동시에 전이관계에서의 통신 이벤트나 상태모드 간의 제어흐름을 추적하여 실제 실행코드를 생성하는 순서에 대한 정보를 얻을 수 있어서 가능하다[6, 8].

SRL/ATM에서 Ada 실행코드로의 변환과정은 앞서 보인 DoME/ATM으로부터 SRL/ATM 코드로 변환하는 과정과 유사하다. 또한 포함된 각각의 구성요소들의 기능도 유사하다. 실제 Ada 코드 변환기의 변환과정은 SRL/ATM 코드 변환기에서 생성된 SRL 파일을 입력받아 SRL/ATM 분석기를 통해 머신, 모드, 포인트, 포트에 해당하는 각각의 정보와 전이에서의 이벤트, 시간제약, 조건에 대한 내용을 추출하고 통신 이벤트나 모드 간의 전이관계를 추적하여 우선순위를 결정한다. 이렇게 추출된 정보와 결정된 우선순위는 SRL/ATM 파스트리 생성기

를 통해 파스트리로 생성된다. SRL/ATM 파스트리로부터 데이터베이스 Unloader를 통해 추출된 정보와 우선순위를 ATM 데이터베이스에 저장한다. 또한 SRL/ATM 파스트리로부터 결정된 우선순위 탐색을 통해 ATM 구성요소들을 추적하여 각 요소에 대한 Ada 구문과 매핑이 이루어진다. Ada 실행코드의 생성은 각 SRL 구문에 대한 Ada 구문을 미리 정의함으로써 이를 이용하여 해당 SRL 구문에 대한 Ada 구문으로 대체할 수 있다. Ada 실행코드의 생성순서는 통신 이벤트나 모드 간 전이관계를 통해 결정된 우선순위 탐색에 의해 결정된다.

다음 [그림 4]는 SRL/ATM에 대한 Ada 실행코드 변환기의 동작과정을 나타낸다.



[그림 4] SRL/ATM에 대한 Ada 코드 변환기의 동작 흐름도

4.2.1 SRL/ATM 분석기

SRL/ATM 분석기는 입력된 SRL 파일로부터 검증을 위한 Ada 실행코드 생성과 관련된 정보를 추출한다. 먼저 매개언어인 SRL 파일을 분석하여 머신, 모드, 포인트, 포트의 정보와 모드내 Element의 정보를 추출한다. 또한 전이관계에서 임의의 동작이나 행위를 나타내는 이벤트나 시간제약, 조건에 대한 정보 추출과 통신 이벤트나 모드간 제어의 흐름을 통해 동작 우선순위를 결정한다.

4.2.2 SRL/ATM 파스트리 생성기

SRL/ATM 파스트리 생성기는 각 구성요소로부터 추출된 정보와 결정된 우선순위의 정보를 가지고 파스트리를 생성하는 도구이다. DoME/ATM 파스트리에서 각 노드는 명세단위인 그래프, 머신, 모드, 포인트, 포트가 된다. 이는 ATM 그래프는 머신과 포트의 집합이고, 머신은 모드와 포인트의 집합이라는 ATM의 기본정의에서 유도된다. 또한 DoME/ATM 파스트리에서 ATM 명세의 블록구조, 계층구조, 명세에서 나타난 전이에 따른 동작의 흐름 등도 분석할 수 있다. 이러한 DoME/ATM 파서트리의 특성은 그대로 SRL/ATM 파스트리 생성 시에도 유지되어 SRL/ATM 파스트리에서도 계층구조 생성 및 전이의 동작흐름에 따른 우선순위를 결정할 수 있다.

4.2.3 SRL/ATM 파스트리 Unloader

SRL/ATM 파스트리 Unloader는 SRL/ATM 파스트리 생성기를 통해 생성된 파스트리로부터 이미 정의된 SRL 구문에 대한 해당 Ada 구문으로 코드를 변경하여 Ada 파일을 생성하는 도구이다. 파스트리 Unloader는 결정된 동작우선순위의 탐색에 따라 각 구성요소에 대한 SRL 구문을 Ada 구문으로 대체한다.

4.2.4 데이터베이스 Unloader

데이터베이스 Unloader는 SRL/ATM으로부터 Ada 실행코드를 생성하는데 필요한 머신, 모드, 포인트, 포트에 대한 추출된 정보와 통신 이벤트나 모드 간의 전이관계에서의 동작 흐름을 통해 결정된 우선순위에 대한 정보를 ATM 데이터베이스에 저장한다. 이렇게 저장된 정보는 Ada와 같은 실행코드를 생성할 때 기본적인 자료로 제공된다.

5. 결론 및 향후 연구과제

본 논문에서는 ATM으로 명세된 실시간 시스템에 대한 재/역공학 측면에서의 개발 및 검증을 위한 코드 변환기를 설계하였다. 또한, DoME을 이용하여 ATM 명세도구를 개발하고 이를 이용하여 실시간 시스템의 특정 요구사항에 대한 ATM을 명세하였다. ATM은 DoME을 이용한 그래픽 표기법에 바탕을 두고 있기 때문에 기존의 정형기법에 비하여 높은 이해도 및 표현력을 제공하며, 구조적이고 모듈화된 명세를 지원한다.

명세된 ATM은 DoME/ATM 스크립트 파일로 저장되고 이에 대한 명세분석을 통해 정보를 추출하여 다른 분석도구가 이용할 수 있도록 ATM DB에 저장하거나 효율적 코드 생성을 위한 매개언어인 SRL로 변환하였다. 이러한 SRL로부터 SRL/ATM 분석기를 통해 실행코드와 관련된 정보를 추출하여 실시간 시스템의 개발 및 검증을 위한 Ada 코드 변환기를 설계하였다.

향후의 연구과제로는 Ada 코드 변환기에 대한 설계를 바탕으로 순공학 과정에서의 요구명세로부터 효율적으로 코드를 생성하기 위한 코드변환기 구현 즉, 매개언어인 SRL에 대한 분석을 통해 정적 정보와 동적 정보를 추출하여 다양한 실행코드를 생성할 수 있는 코드 변환 시스템 개발이 요구된다. 또한 역공학 측면의 검증을 위한 Ada와 같은 기존 프로그래밍 언어로부터 ATM과 같은 명세도구로의 역변환시스템 개발과 아울러 실제 개발된 ATM과 같은 정형기법들을 시스템 개발에 효율적으로 적용하기 위한 다양한 자동화 도구들의 개발이 요구된다.

참고문헌

[1] A. Shaw, *Communicating Real-Time State Machines,* IEEE Transactions on Software Engineering, Vol. 18, No. 9, pp. 805-816, September 1992

[2] C. A. R. Hoare, *Communicating Sequential Processes,* Prentice-Hall, 1985

[3] Honeywell, DoME Guide, Honeywell, Inc., 1999

[4] C. Heitmeyer and D. Mandrioli, Formal Methods for Real-Time Comouting: An Overview, John Wiley and Sons Ltd., 1995

[5] D. Harel, *Statecharts: A Visual Formalism for Complex System, Science of Computer Programming,* Vol. 8, pp. 231-274, 1987

[6] Feldman and Koffman, *Ada95*, Addison-Wesley, 1996

[7] I. Kang and I. Lee, *State Minimization for Concurrent System Analysis Based on State Space Exploration, Proceedings of Conference on Computer Assurance*, Gaithersburg, June 1994

[8] S. jahanian and A. Mok, *Modechart : A Specification Language for Real Time Systems,* IBM Technical Report: RC 15140, November 1989

[9] Sitaram C. V. Raju, An Automatic Verification Technique for Communicating Real-Time State Machines, Dept.of CS&E at University of Washington, TR 93-04-08, 1993

[10] 노경주, 박지연, 이문근, 실시간 시스템의 순환공학을 위한 정형기법 : 추상 시간 기계, 한국정보과학회 학술발표논문집(A), 27권 1호, pp. 477~479, 2000