실시간 객체 지향 모델을 지원하기 위한 정형 명세

정유성* 최수진* 강인혜** 양승민**
{qaze, sjchoi}@realtime soongsil ac.kr {kang, yang}@computing soongsil.ac.kr
*숭실대학교 대학원 컴퓨터학과 **숭실대학교 정보과학대학 컴퓨터학부

Formal Specification for Real-Time Object-Oriented Model

Youseong Jeong* Soojin Choi* Inhye Kang** Seungmin Yang**
*Dept. of Computer Science, Soongsil University **School of Computing, Soongsil University

요 약

실시간 시스템은 응용분야의 특성상 높은 신뢰성을 요구하므로 설계시 시스템의 정확성과 안전성을 보장하는 것은 매우 중요하다 신뢰성 보장을 위한 방법으로 정형기법(formal method)을 이용한 명세 방법이 연구되어 왔다 정형적인 명세를 사용하는 경우 원하는 시스템의 특성에 대한 검증이 가능하며 자연어로 명세한 경우보다 모호함이 줄어들어 의사소통을 하는데 있어서 명확성을 제공한다 그러나 이런 장점에도 불구하고 객체 지향 개발 방법론에서의 정형적인 명세에 관한 연구가 미흡하다

본 논문에서는 객체 지향적 모델을 기반으로 하는 실시간 시스템을 위한 정형명세 언어인 Timed State Chart(TSC)를 제안한다 TSC는 Statecharts와 같은 계층적 상태 기계 모델(state machine model)로서 다양한 시간 제약사항의 명세를 위해 클럭 변수(clock variable)를 도입하여 실시간 객체(Real-Time Object 또는 RTO)를 명세한다 TSC를 이용하여 기존의 연구에서 표현할 수 없었던 주기와 마감시간과 같은 실시간 시스템의 다양한 요구사항을 효과적으로 표현할 수 있다.

1. 서 론

시간적 제약성과 논리적 정확성을 동시에 요구하는 실시간 시스템은 산업체의 공정 제어 및 자동화, 비행기의 관제 및 제어, 원자력 발전소의 통제시스템과 같은 분야에서 널리 사용되어 왔다 근래의 반도체 및 통신망 같은 컴퓨터 관련 제조 기술의 급격한 발전과 정보화 사회에 대한 시대적 요구에 힘입어 실시간 시스템은 그 응용분야가 멀티미디어 초고속 통신, 인공지능 로봇, 무인자동차, 첨단 군용무기등과 같이 다양하게 확장되어지고, 그 규모와 복잡도 또한 증가하고 있다

이러한 응용분야에서의 실시간 시스템은 다음과 같은 두 가지 사항을 요구한다 첫째 개발 생산성의 향상과 유지보수 및 재사용의 효율향상이 요구된다 많은 이들이 이와 같은 요구사항을 해결하기 위하여 객체 지향 기술을 실시간 시스템에 접목시키고자 하였다 대표적인 예로서 Real-Time UML, TMO, dRTO가 있다 Real-Time UML[1][2]은 UML에 실시간 특성을 추가하여 확장한 모델로서 다양한 관점에서 객체를 설계할 수 있는 뷰(view)를 제공한다 그러나 객체의 행위를 나타내는 상태 도표(state diagram)는 시간에 따른 상태 변화에 대한 명세가 부족하다 TMO[3] 와 dRTO 모델[4]은 메소드들을 구동 특성에 따라 시간에 의한 구동 메소드와 메시지에 의한 구동 메소드로 구분하고 있다. TMO는 기본 병행성 제약을 두어 설계 시 시간 검증이 가능하게 하여 보다 높은 예측력을 갖는다 그러나 C'이라는 구조화된 영어로의 명세는 시스템의 계층적 분석과 검증을 힘들게 한다 dRTO는 실시간 객체 지향 모델로서 다양한 시간 제약사항 명세뿐 아니라 신뢰도를 높이기 위한 결함 감지 및 처리 개념을 포함하고 있다

둘째 실시간 시스템은 그 응용분야의 특성상 높은 신뢰성을 요구한다 신뢰성 보장을 위해 시스템은 개발단계의 초기에서부터 정확히 명세되어야 하고, 시스템의 특성이 미리 검증되어야 한다 이와 같이 엄격한 설계와 검증을 요구하는 실시간 시스템을 정확히 명세하기 위해서 근래에는 정형적인 방법을 많이 이용한다. 대표적인 예로서, Harel이 제안한 Statecharts[5][6]가 있으며 이것은 상태 기계의 계층적 모듈별 분석이 가능하기 때문에 대형 시스템을 표현하는데 간결한 방법을 제공한다 이와 같은 이유로, Statecharts는 Real-Time UML, Modechart에서 수정 및 확장되어 사용되었다 그러나 timeout과 같은 이벤트를 통해 기본적인 시간 제약사항을 표현할 수 있지만 주기와 마감시간 혹은 시간의 흐름에 따른 시스템의 상태 변화를 표현하기 힘들다 Modechart[7]는 시스템의 계층적 분석이 가능하며 주기적·돌발적 작업의 명세가 가능하나 전이에 관한 시간 제약사항이 하한값과 상한값으로 표현되기 때문에 복잡한 시간 제약사항의 명세가 힘들다. Timed Automata[8]는 클럭(clock)이라는 개념을 도입하여 다양한 표현력을 제공하는 정형화된 언어로서 실시간 시스템의 요구사항들을 효과적으로 명세할 수 있다. 그러나 시스템의 계층적 분석에 관한 표현이 없으며 다양한 시간 제약사항의 표현은 시스템의 분석을 힘들게 한다

위의 두 가지 요구사항을 해결하기 위한 기존의 연구들은 별개로 진행되어 왔다 이런 이유로 객체 지향 실시간 시스템의 정형적인 명세 방법에 관한 연구가 부족하였으며 실시간 객체와 주기 및 마감시간과 같은 실시간 시스템의 다양한 요구사항들을 반영할 수 없었다

본 논문에서는 dRTO 모델을 비롯한 객체 지향 기반의 실시간 시스템을 위한 정형 명세 언어인 Timed State Chart(TSC)를 제안한다 TSC는 Statecharts와 같은 상태 기계 모델로서 시스템을 구성하는 실시간 객체 단위로 명세된다 실시간 객체는 각각의 메쏘드에 해당하는 하위상태를 가진 병렬상태로 표현되며 메쏘드의 주기와 마감시간을 시간-구동 순차상태와 메시지-구동 순차상태를 통하여 나타낸다 실시간 시스템의 다양한 시간 제약사항과 시간의 흐름에 따른 실시간 객체의 상태 변화에 대한 표현하기 위하여 클럭 변수를 도입한다 TSC는 실시간 시스템의 시뮬레이션, 코드 생성 그리고 실시간성 스케쥴링가능성 검증 도구의 개발을 위한 실행 모델을 제시한다

본 논문은 다음과 같이 구성된다 2장에서는 실시간 객체 모델인 dRTO 모델에 대해 설명한다 특히, 실시간 객체에 어떠한 실시간 특성이 있는지 설명한다 3장에서는 실시간 시스템을 위한 정형화된 명세 언어인 TSC를 제안하고, dRTO 모델을 기반으로한 실시간 객체의 행위를 명세하는 방법을 제안한다 마지막으로 4장에서 결론 및 향후 연구방향으로 마무리 한다

2. dRTO 모델

dRTO 모델은 실시간 객체 지향 모델로서 실시간 시스템의 다양한 시간 제약사항 명세뿐 아니라 신뢰도를 높이기 위한 결함 감지 및 처리 개념을 포함하고 있다[4] dRTO 모델에서 실시간 시스템은 RTO들의 집합으로 정의된다 하나의 RTO는 그 객체에서 사용되는 자료들과 그 객체의 기능을 담당하는 실시간 메쏘드들로 구성된다 실시간 메쏘드는 구동 특성에 따라 시간-구동 메쏘드(Time-triggered Method 또는 TM)과 메시지-구동 메쏘드(Message-triggered Method 또는 MM)로 구분된다. TM은 한 번 생성된 쓰레드가 명세된 주기마다 활성화되어 작업을 수행한다 MM은 각각의 메쏘드 호출에 대해 새로운 쓰레드로 생성되어 작업을 수행한다 RTO간의 통신은 MM을 호출함으로써 이루어지며, 호출자가 반환값을 기다리는 동기적 호출방법과 호출자가 호출 후 바로 다음 동작을 수행하는 비동기적 호출방법이 있다

dRTO 모델에서는 시스템의 다양한 시간 제약사항을 명세할 수 있다. 첫째, TM은 주기와 마감시간을 명세하여 각 주기마다 작업을 수행

* 본 연구는 한국과학재단 목정기초 연구과제 지원(과제번호 96-0101-07-01-3)으로 수행되었습니다

하고 주어진 마감시간 내에 작업을 끝내도록 한다 둘째, MM에는 주어진 작업의 마감시간을 명세할 수 있다 셋째, 동기적 호출방식의 메시지에 마감시간을 지정함으로써 이 메시지에 의해 구동하는 MM이 마감시간 내에 결과를 반환하도록 한다 마지막으로 각 메쏘드의 행위는 TSC를 사용하여 명세함으로써 다양한 실시간 제약사항을 나타낼 수 있다 TSC에 대해서는 다음 장에서 자세히 설명한다

3. Timed State Chart

Timed State Chart(TSC)는 dRTO 모델을 비롯한 객체 지향 기반의 실시간 시스템의 행위를 명세하기 위한 정형 명세 언어이다 이 장에서는 TSC의 문법과 그 의미를 살펴보고 TSC를 사용하여 dRTO 모델의 행위를 어떻게 명세하는 지 설명한다

3.1 계층적 상태 기계 모델

TSC는 Statecharts[5]와 같이 계층적 상태 기계 모델로서 상태와 전이로 구성된다 상태의 계층적인 구조는 시스템의 계층적 설계 및 분석이 가능하게 한다(그림 1) 상태는 시스템 제어가 머무르며 정의된 액션을 수행하는 위치로서, 상태 이름, 변수, 액션을 포함한다 변수에는 일반 데이터 변수와 시간 명세를 위한 클럭 변수가 있으며 한 상태에서 정의된 변수는 그 하위상태에서 사용될 수 있다 액션은 그 상태에서 수행해야 할 동작을 나타낸다

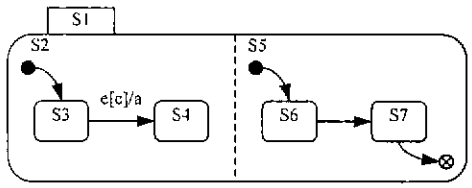


그림 1 Timed State Chart(TSC)

상태는 그 의미에 따라 병렬상태와 순차상태로 구분된다.(Statecharts에서는 AND-state와 OR-state로 표현한다) 병렬상태는 동시에 모든 하위 상태에 시스템 제어가 머무르며 각 하위상태의 액션을 동시에 수행할 수 있다 순차상태에서는 한 시점에서 그 하위 상태들 중 단 한 상태에만 시스템 제어가 머무를 수 있다 따라서 순차상태내의 하위 상태의 액션들은 순차적으로 실행된다 그림 1의 상태 S1은 병렬상태로 하위상태인 S2와 S5가 동시에 실행되고, 상태 S2(또는 S5)는 순차상태로 하위상태 S3와 S4(또는 S6와 S7)가 순차적으로 실행된다.

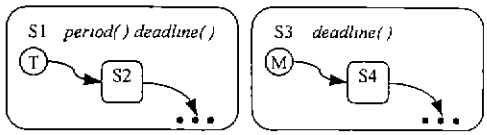
상태 변화를 나타내는 전이는

시작상태, 전이 이벤트, 전이 조건문, 치환문, 다음상태

으로 구성되며 시간을 소모하지 않는 것으로 가정한다 그림 1에서 제어가 시작상태 S3에 있고, 전이 이벤트 e가 발생하고 조건 c가 참일 경우 전이가 실행될 수 있다 이벤트는 어떤 사건이 일어난 것을 나타내는 시그날로서 외부 시그날이나 통신 시그날 등이 있다. 전이가 실행되면 치환문 a에 따라 변수의 값을 변경하고 다음상태 S4로 제어가 이동한다 만약 조건문이 없으면 참이라고 간주한다

이와 같은 일반적인 전이와 달리 시작상태를 가지고 있지 않는 특수한 전이를 초기전이라 한다 초기전이는 하나 이상의 하위상태로 구성되어 있는 순차상태 내에서 어떤 하위상태에서 시작할 지를 나타내며 그림 1과 같이 시작상태를 ●으로 표현한다 제어가 순차상태에 들어오면 그 안의 초기전이가 실행된다. 이와 반대로 전이의 다음상태가 없는 특수한 전이를 종료전이라 한다 종료전이는 시스템이 이 상태에서의 실행을 완료했음을 의미하며 그림 1과 같이 다음상태를 ⊗으로 표현한다

TSC는 dRTO 모델을 지원하기 위해 시간-구동 순차상태와 메시지-구동 순차상태를 추가한다 TM을 위한 시간-구동 순차상태는 주기 p와 마감시간 d를 포함한다. 초기전이는 최초의 쓰레드 생성시에만 발생하며 이후 명세된 주기마다 쓰레드 활성화를 위하여 초기상태로 제어가 이동한다 활성화된 쓰레드는 마감시간 이전에 작업이 완료되어야 한다(단 $p \geq d$ 라고 제한한다) MM을 위한 메시지-구동 순차상태는 마감시간 d를 포함하며 메시지에 의한 호출에 의해 초기전이가 발생한다 초기전이의 발생은 MM의 새로운 쓰레드의 생성을 의미하며 생성된 쓰레드는 마감시간 이전에 작업이 완료되어야 한다 그림 2에서와 같이 초기전이의 시작을 시간-구동 순차상태에서는 T로 메시지-구동 순차상태에서는 M으로 표시한다



(a) 시간-구동 순차상태 (b) 메시지-구동 순차상태

그림 2 시간-구동 순차상태와 메시지-구동 순차상태

3.2 시간 제약사항

TSC에서는 시간 제약사항을 다양한 방법으로 명세할 수 있다 앞서 언급한 바와 같이 시간-구동 순차상태와 메시지-구동 순차상태에 주기와 마감시간을 명세함으로써 상태에 시간을 연관시킬 수 있다 또한 다음과 같이 액션에 최소 실행시간, 최대 실행시간을 명세할 수 있다

action α[min,max]

TSC에서는 보다 다양한 실시간 제약사항을 표현하기 위해 Timed Automata[8]에서 제안한 클럭 변수(clock variable)를 사용한다 클럭 변수는 dense-time모델을 사용하여 0을 포함한 양의 실수의 집합 R 을 시간 영역으로 갖는 변수이다 모든 클럭 변수는 초기값 0부터 일정한 비율로 증가하며 어느 한 순간에 클럭 변수가 지니고 있는 값은 가장 최근에 0으로 지정한 이후의 경과 시간과 동일하다 클럭 변수는 전이시 치환문에 의해서 값이 0으로 지정될 수 있고, 조건문에서는 전이가 클럭 변수의 현재 값이 제약사항을 만족하는 경우에만 실행되도록 할 수 있다 우리는 이러한 제약사항들을 클럭 제약사항(clock constraint)이라 부른다 클럭 제약사항 δ는 다음과 같이 정의된다

$$\delta := x \leq c \mid c \leq x \mid x = c \mid \neg\delta_1 \mid \delta_1 \wedge \delta_2$$

여기에서 c는 양의 유리수이고 x는 클럭 변수이다

그림 3은 클럭 변수로 시간 제약사항을 어떻게 표현하는지 보여준다 (a)는 timeout을 나타낸다 S1으로의 진입 시 치환문에서 클럭 변수 x를 0으로 초기화하고 전출 시 조건문은 x=3으로 표현하면 시스템은 3 단위시간 동안에만 S1에 위치한 후 S2로 이동할 수 있다 (b)는 delay를 표현한다 시스템은 S2에 진입한지 5 단위시간이 지난 후 S3로 이동할 수 있다 (c)는 2개의 클럭 변수를 사용하여 여러 전이 사이의 시간관계를 나타낸다 두 클럭 변수 x와 y가 서로 다른 값을 가지며 독립적으로 작용하는 것을 볼 수 있다 이와 같이 클럭 변수는 다양한 시간 제약사항을 표현할 수 있다

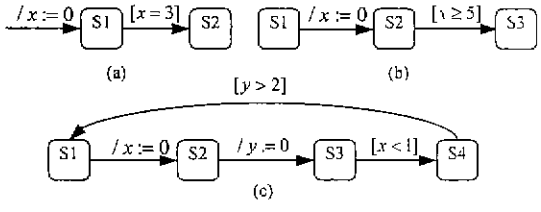


그림 3 클럭 변수의 예

3.3 실시간 객체간의 통신

TSC에서 프로세스간의 통신은 호출자(caller)와 피호출자(callee) 사이의 채널을 통해 이루어진다 호출자는 해당하는 채널에 대한 호출 액션을 수행함으로써 통신을 개시할 수 있다 호출 액션은 다음과 같다

channelname!(argument-list)

호출 액션을 수행하면 이 액션에 관한 이벤트가 발생한다 피호출자는 다음과 같은 전이 이벤트를 사용하여 호출의 발생을 기다린다

channelname?(argument-list)

예를 들어, 그림 4에서 제어가 각각 S2와 S4에 있을 때 S2에서 액션 ch!(10)을 수행하면, S4에서 S5로의 전이가 발생하여 변수 x에는 10이 지정되고 제어가 S5로 이동한다

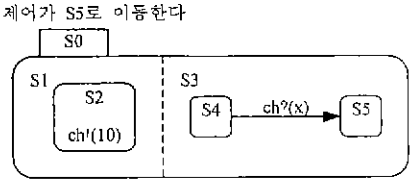


그림 4 두 프로세스간의 통신

dRTO 모델에서의 메소드 호출을 TSC로 표현하기 위하여 호출자와 피호출자간에 다음과 같은 채널을 사용한다

- Caller_RTO_name Method_name Callee_RTO_name Method_name – 호출자 RTO가 통신을 개시하기 위해서 사용되는 채널로서 원하는 메시지를 피호출자에게 넘겨준다
- Callee_RTO_name Method_name Caller_RTO_name Method_name – 피호출자가 반환값을 호출자에게 넘겨주기 위한 채널이다

이 두 채널을 사용하여 비동기 호출방식과 동기 호출방식을 모두 지원할 수 있다 비동기 호출방식과 동기 호출방식의 호출은 모두 다음과 같은 호출자의 액션과 호출자의 전이 이벤트로 표현될 수 있다

Caller_RTO_name Method_name Callee_RTO_name Method_name!(argu_list)

Caller_RTO_name Method_name Callee_RTO_name Method_name?(argu_list)

동기적 호출방식은 호출 후 피호출자가 반환하는 값을 기다려야 하기 때문에 호출자는 다음과 같은 전이 이벤트를 사용하고

Callee_RTO_name Method_name Caller_RTO_name Method_name?(recv)

피호출자는 다음의 액션을 취함으로써 프로세스간의 통신을 완료한다

Callee_RTO_name Method_name Caller_RTO_name Method_name!(return_value)

3.4 실행 모델

TSC로 명세한 시스템의 행위는 환경으로부터의 외부 자극에 대한 시스템의 반응을 표현하는 가능한 실행(execution 또는 run)들의 집합이다 실행은 시스템의 상황을 나타내는 일련의 단편(snapshot)들로 구성된다 이런 단편을 status라 부른다 초기 status가 있고 후속된 status는 앞선 status가 step을 실행함으로써 나타난다(그림 5)

status는 앞으로의 행위에 영향을 미치는 정보로서 다음과 같다

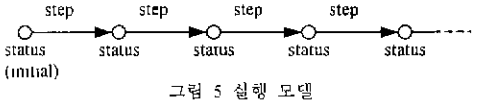


그림 5 실행 모델

- configuration 제어가 동시에 위치할 수 있는 상태들의 집합
- 이벤트 집합 앞으로의 행위에 영향을 미치는 이미 발생한 이벤트들의 집합
- 데이터 변수의 값
- 클럭 변수의 값
- 활성화된 액션들의 집합 현재 수행을 시작하는 또는 수행되고 있는 액션을 말하며 액션의 이름과 지금까지 실행된 시간의 순서쌍 (α, t)으로 나타난다
- 활성화된 전이들의 집합 전이 이벤트는 이벤트 집합에 나타나고 전이 조건이 참인 전이들로 실행이 가능한 전이들의 집합

만약, configuration에는 병렬상태인 s가 있다면 항상 s의 하위상태 역시 configuration에 포함된다

초기 status는 다음과 같이 주어진다

- 초기 configuration 시스템에 해당하는 상태(최상위 상태)를 포함하는 최소 configuration
- 이벤트 집합 ∅
- 데이터 변수 값 초기값 또는 그 변수값 영역 내의 임의의 값
- 클럭 변수 값 0으로 초기화
- 활성화된 액션들의 집합 초기 configuration의 상태 s가 액션 α를 포함하면 (α,0)가 이에 속함
- 활성화된 전이들의 집합 초기 configuration의 순차 상태 s의 실행 가능한 초기전이를 포함

step은 status의 변화를 말하는데 다음과 같은 두 가지의 step이 있다.

- instantaneous step 이벤트 발생이나 전이의 실행에 status의 변화
- timed step 시간의 흐름에 따른 status의 변화

현재 status를 St_i, 다음 status를 St_j이라고 하자 i=1,2일 때, St_i은 configuration C_i, 이벤트 집합 E_i, 데이터 변수의 값 D_i, 클럭 변수의 값 X_i, 활성화된 액션들의 집합 A_i, 활성화된 전이들의 집합 T_i로 구성된다 현재 status St_1에서 가능한 step은 다음과 같다

첫째 액션의 수행 완료에 따른 step으로, 현재 status의 액션 집합에 속한 (α, t)가 $min(\alpha) \le t \le max(\alpha)$이면 액션 α의 수행이 현 시점에서 끝날 수 있다 이 instantaneous step에 따른 다음 status 변화는 다음과 같다 $C_2=C_1$, $X_2=X_1$이고 E_2는 $E_1 \cup \{end(\alpha)\}$이다 D_2는 α가 데이터 변수 값을 바꾸는 액션이면 α에 의해 데이터 값을 적절히 변경된다 A_2는 A_1에서 (α, t) 제거하고 end(α)나 변수 값 변경에 의해 활성화된 전이가 존재하면 T_2에 추가한다

둘째, 활성화된 전이의 실행에 따른 step으로, T_1의 활성화 전이 t=(s, e, c, a, s')의 실행에 의해 다음과 같이 status가 변한다 C_2은 C_1에서 s와 그 하위 상태 제거하고 s'과 그와 연관된 병렬 하위 상태를 모두 추가한다 E_2는 전이 t에 해당하는 이벤트 e는 E_1에서 제거하고, 상세된 상태에서 정의된 액션 α_1이 있으면 stopped(α_1)를 추가한다 s' 에 액션 α_2를 정의하고 있으면 E_2에 start(α_2)를 추가한다 D_2와 X_2는 전이에 의해 바뀌는 변수 값을 변경한 것이고, A_2는 A_1에서 제거된 상태에 정의된 액션에 해당하는 부분 제거하고, 첨가된 상태에 해당하는 액션 α_2에 대해 (α_2, 0)를 포함시킨다 T_2는 전이에 의해 조건이 만족하지 않는 전이는 제거하고 활성화된 전이를 추가한다

셋째, 시간의 흐름에 의한 timed step이 있다 다음 조건들을 모두 만족한 상태에서만 시간이 d 만큼 흐를 수 있다

- 현 활성화 전이중 d 시간 이후에 전이 조건의 변화에 의해 비활성화 되는 것이 없을 경우
- 모든 활성화 액션 (α, t)에 대하여 $d \le max(\alpha)-t$를 만족할 경우

이 때 모든 클럭 변수의 값은 d 만큼 증가하고, 액션 리스트에 있는 실행된 시간이 d 만큼 증가한다 그리고 클럭 변수 값 변경에 따른 활성화 전이가 바뀔 수 있다

4. 결론 및 향후 연구방향

본 논문에서는 실시간 객체 모델인 dRTO를 비롯한 객체 작향 기반의 실시간 시스템의 행위를 명세하기 위한 정형 명세 언어인 TSC를 제안하였다 TSC는 실시간 시스템을 구성하는 실시간 객체 단위로 명세된다 실시간 객체는 자신의 메소드들에 해당하는 상태들을 가진 병렬상태로 표현함으로써 실시간 시스템의 병행성(concurrency)을 나타낼 수 있었다 TSC는 기본적으로 계층적 상태 기계 모델이기반 실시간 객체의 TM과 MM을 지원하기 위한 시간-구동 순차상태와 메시지-구동 순차상태를 추가하여 기존의 연구에서 표현할 수 없었던 주기의 마감시간을 효과적으로 표현할 수 있다 클럭 변수를 도입하여 실시간 시스템의 다양한 요구사항과 시간의 흐름에 따른 실시간 객체의 상태 변화에 대한 표현이 가능하다 실시간 객체 간의 통신을 위해 채널을 설정함으로써 명세 과정에서 발생할 수 있는 오류를 감소시킨다 그리고 실시간 시스템의 시뮬레이션, 코드 생성 그리고 검증 도구의 개발을 위한 실행 모델을 제시하였다

현재 TSC에서 제시한 실행 모델을 바탕으로 명세한 시스템의 안전성과 실시간 제약사항등을 검증하기 위해 reachability analysis와 model checking방법에 관한 연구가 진행중에 있다 향후 IDEEA(Intergrated Development Environment for Embedded Application) 프로젝트의 일환으로 사용자의 요구사항을 분석할 요구사항 분석기, 분석된 사용자 요구사항과 TSC로 명세한 시스템의 일관성을 검사할 검증기, 그리고 시뮬레이터와 코드 생성기등의 도구에 대한 연구가 진행되어야 한다

5. 참고문헌

[1] Bran Selic, ObjecTime Limited Jim Rumbaugh, "Using UML for Modeling Complex Real-Time Systems", Rational Software Coporation Mrach 11, 1998

[2] Grady Booch , "Unified Modeling Language for Real-Time Systems Design", http://www.rational.com

[3] K.H Kim, "Object Structures for Real-Time Systems and Simulators", IEEE Computer, 1997

[4] 이신, 손현수, 양승민, "실시간 객체 모델 dRTO" 한국정보과학회 논문지 심사중

[5] D.Harel and A.Naamad, "The STATEMATE Semantics of Statecharts", ACM Transactions on Software Engineering and Methodology, Vol 5, No.4, Octorber 1996, pp 293-333

[6] D.Harel and Eran Gery, "Executable Object Modeling with Statecharts", IEEE Computer, Vol 30, No 7. July 1997

[7] Farnam Jahanian, Aloysius K Mok "Modechart A Specification Language for Real-Time System", IEEE Transactions on Software Engineering 1994

[8] R Alur and D L Dill, "A Theory of Timed Automata" Theoretical Computer Science 126 183-235, 1994