GCSR 의 UML-RT 언어로의 제안을 위한 STATECHART 와의 비교

[0]김진현*,박명환*,최진영*,강인혜**
고려대학교 컴퓨터학과*
숭실대학교 컴퓨터학부**

Comparison with STATECHART for Proposing GCSR as UML-RT Language.

[0]Jin-Hyun-Kim*, Myung-Hwan-Park*, Jin-Young -Choi*, Inhye-Kang**
Department of Computer Science and Engineering, Korea University*
School of Computing, Soongsil University**

요 약

본 논문은 UML-RT 도구로서 GCSR(Graphical Communicating Shared Resource)를 제안하는데 있어 기존의 UML 의 기본 다이아그램 중 하나인 Statechart 와 비교 분석함으로 그 기능과 장 단점을 제시한다 기존의 정형 명세 언어인 Statechart 에서는 실시간 시스템의 명세에 있어 필수적인 시간적인 개념과 우선순위 개념이 세련적이다 그러나 정형명세의 또 다른 언어인 GCSR 이 가진 시간적 개념과 우선순위 개념의 효용을 보이고 이를 Statechart 의 실시간 시스템의 명세와 비교함으로 UML-RT 로서의 GCSR 을 제안한다

1. 서론

UML(Unified Modeling Language)은 복잡한 시스템의 구조와 판계를 표현하는 언어이다

대부분의 실시간 시스템은 병렬적이면서 복잡한 구조를 지니고 있기 때문에 이를 설계하는데 있어 새로운 도구로 UML 제시되고 있으며 이를 위해 여러 방법론과 언어들이 제시되고 있다 하지만 실시간 시스템은 시간적 제약과 공유 자원 관리를 위한 우선 순위 개념으로 인해 기존의 UML 의 언어로 알려져 있는 Statechart[2]니 Activity chart 가 지닌 문법으로는 많은 제약과 불편이 따른다 본 논문에서는 시간 개념과 명확한 우선순위 개념을 지닌 도식적 명세 언어인 GCSR(Graphical Communicating Shared Resource)을 소개하고 실시간 시스템을 명세 하는데 있어 GCSR 이 지닌 문법과 Statechart 가 지닌 문법 비교 분석함으로 UML-RT 의 언어로 GCSR 을 제안하는 바이다

본 논문은 다음과 같이 구성되어 있다

2 장에서는 UML 에 대한 개념과 UML 의 언어인 Statechart 그리고 본 논문에서 제안하는 GCSR 를 소개하고, 3 장에서는 두 언어가 가진 실시간 시스템을 명세하기 위한 시간적인 특징을 살펴보겠다 4 장에서는 두 언어로 명세한 실시간 시스템의 예를 살펴보고, 5 장에서는 두 언어의 실시간 시스템의 명세 능력을 비교 분석하고 UML-RT 의 언어로서 GCSR 을 제안한다. 마지막 6 장에서는 결론 및 향후 연구 방향 대해 설명하겠다

2. Statechart 와 GCSR

2.1 UML

UML 은 시스템의 논리적, 물리적 측면을 표현하기 위한 어휘와 규칙을 제공하는 객체 모델링 언어이다 이는 시스템의 개발과 전개를 위해 모델링 자료를 정확하고 분명하게 작성할 수 있도록 그래픽 심벌을 이용하여 모델링 자료의 시각화를 높이고 시스템의 다양한 측면과 관점에서 모델링하기 위해 다이아그램을 제시한다 따라서 시스템의 특성에 따라 가장 잘 표현할 수 있는 다이아그램을 선정하여 작성하게 된다 UML 은 시스템을 바라보는 관점에 따라 구조적 분야, 동적인 분야, 모델관리분야로 나뉘어지며 실시간 시스템과 관련되어서 시간의 흐름에 따른 시스템의 행위(behavior)를 나타내는 동적인 분야의 관점에서 전개된다 동적인 분야의 다이아그램은 시스템의 상태에 초점을 둔 상태 차트 다이아그램 활동 관점에 초점을 둔 액티비티 다이아그램, 시스템의 각 부분의 상호 작용에 초점을 둔 시퀀스(sequence) 다이아그램이나 콜라보레이션(collaboration) 다이아그램으로 나뉘어진다.[1]

본 논문에서는 시스템의 상태에 초점을 둔 상태 차트 다이아그램의 표준으로 제시되고 있는 Statechart 에 중점을 두고 비교할 것이다

2..2 Statechart

Statechart[2]는 관련된 이벤트가 발생할 때, 상태 전이를 표현한 것으로 상태(State)와 전이(Transition) 액션(Action)으로 이루어진다 상태는 특정 시간동안 유지되고 구별되는 상태를 가리키고 전이는 이벤트에 응답하여 객체가 상태를 변화시키는 것 방법을 의미하며 액션은 어떤 상태로 진입 되거나 진출할 때 이벤트가 전이를 발생시키는 것과 같은 단위적인 행동을 말한다

Statechart 는 State diagram 의 골격에다 계층과 동시성 그리고 Broadcasting 의 개념을 더한 것이라 할 수 있다

그림 1 은 간단한 Statechart 를 보여준다 이 다이아그램에서 사각형은 각 상태를 나타내고 화살표는 전이를 나타낸다 Statechart 에서 전이 레이블(label)의 문법은 e(c)/a 이다 e 는 전이를 발생시키는 이벤트이고 (c)는 이벤트가 발생했을 때 전이를 가능하게 하는 조건문이다 a 는 전이가 일어날 때 발생되는 액션을 가리킨다 각 상태간의 계층은 encapsulation 을 사용한다 그림 1 에서 상태 A 와 상태 D 는 서로 지식괴 부모간의 계층관계를 가지고 있고 서로 exclusive-or 관계로 같은 시간에 서로 다른 상태에 제어가 있어서는 안 된다 그림 2 에서 각 상태 붙어있는 작은 화살표는 default 상태가 된다 그림 3 에서는 동시성을 보여주고 있다 즉 가운데 점선은 상태 F 와 G 가 'AND'관계에 있음을 보여준다 즉 상태 F 와 G 가 동시에 있는 것을 의미한다

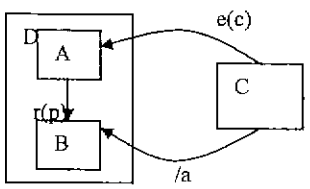

그림 1 Statechart 의 예

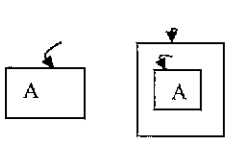

그림 2 default state 의 예

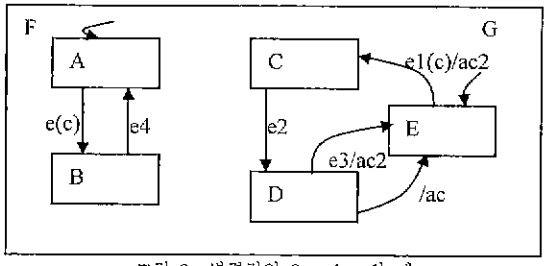

그림 3 병렬적인 Statechart 의 예

2.3 GCSR

실시간 시스템의 잘못된 기능과 관련된 잠재적인 높은 위험성은 실시간 시스템을 구현하기 전에 정확하고 간결하게 명세하고 엄격하게 분석할 수 있는 정형적인 프레임워크를 필요로 한다. 최근에 이르러서는 병렬적인 수행을 표현하는 상태 다이아그램을 제시하고 있는 Statechart 나 Mode-chart, CRSM(Communicating Real-time State Machine)과 같은 몇몇 정형적이고 도식적인 언어가 개발되기에 이르렀다

이 논문에선 소개하는 GCSR(Graphical Communicating Shared Resource)[3]은 기능적이고 자원의 사용을 요구하는 실시간 시스템의 명세와 분석을 위한 정형 명세 언어이다

이것은 다음과 같은 특징을 지니고 있다

첫 번째로 GCSR 은 실시간 시스템을 명세하기 위해 이벤트와 인터럽트, 동시성을 통한 통신 개념을 지원한다 그에 더하여 실시간 시스템의 행위를 올바르게 명세하고 이해하며 수정하기 쉽게 할 수 있는 방법으로 자원의 경쟁을 중재하기 위한 자원에 대한 우선 순위의 명시적인 표현을 제공한다 이것은 기존의 Statechart 나 Modechart, CRSM 이 실제적인 실행 시간 환경을 자원의 경쟁이 없는 것으로 간주하는 것과는 달리 공유된 자원의 스케쥴로 인한 지연에 의해 실시간 시스템이 영향을 받는 다는 것을 가정하고 있다.

두 번째로 GCSR 은 단위적이고 계층적이며 확장 가능한 실시간 시스템 명세를 가능하게 하기 위해 다양한 형태의 노드와 엣지를 제공하며 도식적 모호함과 비구조적 명세를 갖지 않기 위해 Inter-level 전이를 허용하지 않는다

세 번째로 GCSR 은 일반적 연산 가능한 의미론을 지닌 도식적이고 문자적인 언어이다 GCSR 의 의미론은 자원에 대한 우선 순위를 가진 시간 소비의 프로세스 엘지브라인 ACSR(Algebra of Communicating Share Resources)[4]에 의해 제공되기 때문에 GCSR 의 명세는 행위적인 측면에서 요구 사항을 올바르게 충족시키는지 검증할 수 있다

그림 4 에서 간단한 GCSR 의 명세를 볼 수 있다. 원으로 그려진 노드는 시간을 소모하는 자원을 가리킨다 각 엣지에 붙어 있는 레이블은 세가지로 나뉘는데 먼지 아무 레이블도 가지지 않는 레이블과 이벤트를 가리키는 e 와 우선순위 p 를 가지는 (e,p) 레이블, 전이되기 전에 소모해야 할 시간을 가리키는 tm 을 가지는 레이블이며 이것은 자원을 소모하는 노드 뒤에 붙어 있어야 한다 점선으로 된 사각형은 다른 부분에서 제정의 될 수 있다 그리고 검게 칠해진 사각형은 아무 일도 하지 않는 NIL 상태를 가리킨다 그림 5 는 병렬적인 행동을 묘사하는 문법이다

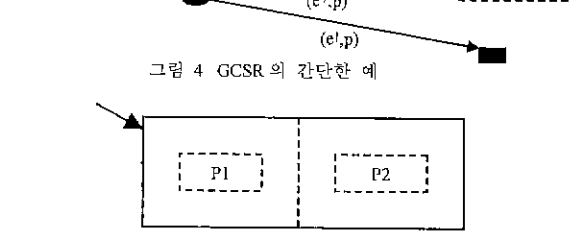

그림 4 GCSR 의 간단한 예

그림 5 병렬적인 GCSR 의 예

3. Statechart 와 GCSR 의 시간 개념

Statechart 에서 시스템의 행동은 외부 환경에 의해 발생되는 일련의 자극에 대한 시스템의 응답을 표현하는 가능한 run 들의 집합이다. 이러한 run 을 이루는 각각의 status 는 step 의 수행에 의해 전행된다 하나의 status 는 활성화된 상태, 행위들, 데이터 변수의 값, 조건 발생된 이벤트, 배정된 행위들, 그리고 시스템의 history 의 정보를 가진다 각 step 의 처음에는 환경이 외부 자극을 주고 전이가 일어나게 되며 전이가 일어남으로 조건 변수와 데이터가 변하게 되고 새로운 이벤트가 발생하게 되고 새로운 활동들이 시작하거나 중지된다 이벤트와 관련되어서는 다음과 같은 가정을 갖는다 내부 및 외부 이벤트에 대한 반응이나 변화는 하나의 step 에서 일어나고 그 step 의 종료 후에야 다른 상태에게 알려진다 또한 하나의 step 에서 유효한 이벤트는 그에 이어지는 다음 step 에까지 기억되지 않는다 하지만 step 과 시간의 관계를 병행적으로 생각하지는 않는다 즉 하나의 step 이 하나의 시간을 의미하지는 않는다는 것이다 따라서 Statechart 에서는 다음과 같은 두 가지 시간과 관련된 모델을 제시하고 있다 첫 번째로 동기적 시간모델(Synchronous Time Model) 은 매 타임 유닛 마다 하나의 step 을 소모한다 즉 이전 step 의 종료 이후의 하나의 타임 유닛 내에, 발생된 모든 외부적인 변화에 대해 반응한다 하지만 비동기적 시간 모델(Asynchronous Time Model)은 하나의 외부적인 변화가 일어날 때마다 반응하고 등시에 여러 개의 변화가 일어나는 것을 허용하며 하나의 시간 유닛 내에 몇 개의 step 의 소모를 허용한다. 하지만 이러한 시간적 모델은 특정 이벤트의 발생을 계수하는 추상적 모델을 제시하는 것이기 때문에 물리적 시간을 의미는 것은 아니며 전통적인 모달 템퍼럴 로직(modal temporal logics)이나 실시간 로직으로는 이러한 추상적 모델에 대해 시간에 대한 양적 특징을 표현하거나 검증할 수 없다 본 논문에서 가정하는 스테이트 차트의 모델은 동기적인 모델을 모델로 삼을 것이다 [5]

ACSR 의 의미론을 따르고 있는 GCSR 에서는 이벤트의 발생은 시간을 소모하지 않는다 따라서 Statechart 에서 언급한 이벤트의 발생은 하나의 step 을 반드시 거쳐야 하는 반면 GCSR 에서는 그렇지 않다 GCSR 에서 시간을 소모하는 경우는 반드시 자원을 소모하는 액션이다. GCSR 에서는 특정 시간 동안 수행을 연기하거나 얼마의 액션이 발생하기까지의 기다리거나 일련의 액션들의 수행이 실행되는 시간을 제한할 수 있다. 또한 주고 받는 이벤트와 취하는 액션마다 공유되는 자원에 대한 우선순위를 가지고 있기 때문에 정확한 스케쥴링에 대한 명세가 가능하다

4. 실시간 시스템의 Statechart 와 GCSR 의 정형 명세의 예

다음의 예는 비행기 이착륙 시, 착륙로와 이륙로 사이, 게이트로 들어가려는 비행기를 위한 임시 정차로에 대한 예이다 이 예에서 비행기는 착륙과 동시에 게이트로 들어가는 것이 아니라 임시 정차로에서 잠시 대기했다가 이륙로를 거쳐 게이트로 들어가게 된다 이 때 이륙로에는 어떠한 이륙하는 비행기도 있어서는 안 된다. 비행중인 비행기는 착륙에 대한 요청을 하고 일정 타임 유닛이 지나면 긴급 착륙 요청을 하게 된다. 이 때 이 임시 정차로를 관리하는 관제소에서는 긴급 임시 정차로를 비우게 되어 비행기가 착륙하게 된다

이륙하는 비행기는 이륙 관제소로부터 이륙 허가를 받게 되는데 이 때 이륙 관제소는 임시 정차로 관제소와 임시 정차로에서 게이트로 들어가는 도로에 대해 경쟁 관계에 있게 된다

그림 6 은 Statechart 로 관제소-착륙 관제소, 이륙 관제소, 임시 정차로 관제소-를 명세한 것이다 이 명세에서는 각 부분의 동시성을 묘사하기는 하지만 여러 대의 같은 일을 하는 비행기가 동시에 같은 요청을 함으로 공유 자원에 대한 스케쥴링 관계는 묘사할 수 없었다

그림 7 은 같은 항공 관제 시스템을 GCSR 로 명세한 그림이다

이 명세에서는 GCSR 이 공유자원에 대한 우선순위 및 정확한 스케쥴링 또한 명세하고 있다 단지 상태의 변화 뿐 아니라 자원에 대한 행위를 묘사하고 있다.

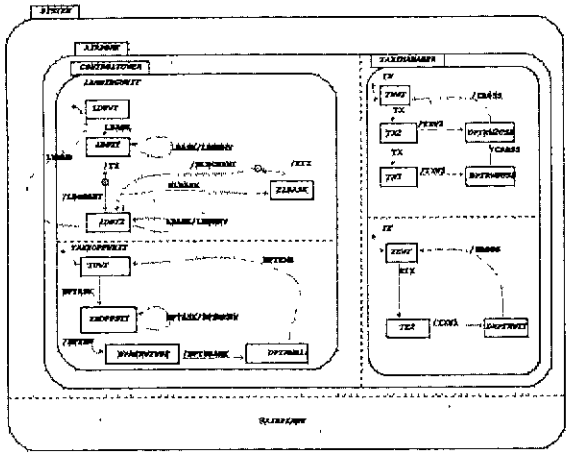

그림 7 : Statechart 로 명세된 관제시스템의 예

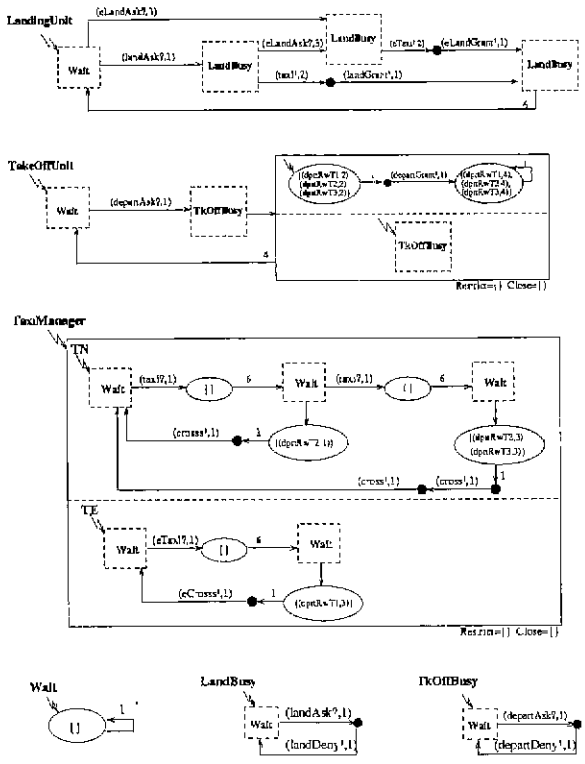

그림 7. GCSR 로 명세된 관제시스템의 예

5. UML-RT 로서의 GCSR 의 제안

실시간 시스템을 명세하는 경우 시간적인 계약은 물론 공유자원을 스케줄링하는 우선순위를 가진 문법이 필요하다. 본 논문에서 UML 의 예로 제시한 Statechart 에서는 단지 상태 변화만을 모사하기 때문에 공유자원을 선점하려는 각 프로세스를 명세하는 자원의 선점에 대한 문법의 부족으로 각 프로세스가 각 자원에 대한 스케줄을 상태로 나타내 주어야 하기 때문에 스테이트머신을 그리기에 쉽지 않다.

하지만 GCSR 의 경우 시간과 자원의 스케줄에 대한 우선순위를 표현하는 다양한 표현력과 문법을 갖추고 있기 때문에 Statechart 보다 실시간 시스템을 표현하는데 더 나은 표현력을 가지고 간결하면서도 분명하게 명세 할 수 있다. 그에 더하여 GCSR 은 ACSR 의 검증 도구와 연동할 수 있기 때문에 실시간 시스템 특히 자원을 선점하려는 병렬적인 프로세스를 가진 시스템이 데드락이 걸리지 않고 올바르게 행동하는지를 행위적인 측면에서 검증할 수 있기 때문에 단지 모의 시험만을 할 수 밖에 없는 UML 의 다른 언어보다 실시간 시스템을 명세 하는데 효과적이고 유용한 언어임을 알 수 있다. 하지만 그러한 GCSR 은 Statechart 보다 다양한 어휘를 가지고 있기 때문에 쉽게 이해하지 못하는 단점도 가지고 있다.

표 1 은 Statechart 와 GCSR 의 명세 및 명세 능력에 대한 비교 분석이다.

	Statechart	GCSR
차트개념	시스템의 상태의 변화묘사	프로세스의 행위묘사
우선순위의 표현력	계층의 차별을 통한 우선순위 표현	명료한 우선순위 표현
자원의 개념	없다	있다
물리적 시간 개념	없다	있다
Inter-level transition	있다	없다
노드의 형태	단순	다양
검증능력	시뮬레이션을 통한 검증	Bi-simulation 을 통한 검증
이해도	쉽다	상대적으로 어렵다

표 1 Statechart 와 GCSR 의 비교

6. 결론

본 논문에서는 기존의 UML 언어인 Statechart 의 실시간 시스템의 명세와 GCSR 의 실시간 시스템의 명세를 비교 분석함으로 UML-RT 의 언어로서 GCSR 을 제시하였다. 본 논문에서 제시한 GCSR 의 표현능력과 ACSR 의 연동을 통한 검증 능력을 볼 때, 실시간 시스템의 명세하기 위한 UML-RT 언어로서의 능력이 충분함을 보였다. 향후 연구 방향으로 기존의 UML 언어와의 연계 방향과 이 언어의 실시간 시스템의 표현에 대한 더 많은 검증이 필요할 것이다.

7 참고문헌

[1] Grady Booch, James Rumbauge, Ivar Jacobson. The Unified Modeling Language User Guide, Addison-Wesley ,1997

[2] David Harel STATECHART A VISUAL FORMALISM FOR COMPLEX SYSTEMS, Science of Computer Programming 8 (1987) pp231-274

[3] Hanene Ben-Abdallah, Insup-Lee , Jin-Young Choi, A Graphical Language with Formal Semantics for the Specification and Analysis of Real-Time Systems H Ben-Abdallah, I Lee and J -Y Choi, *Proceedings of the 16th IEEE Real-Time Systems Symposium*. 1995

[4] I Lee, H Ben-Abdallah, and J -Y Choi, A Process Algebraic Method for Real-Time Systems *Formal Methods for Real-Time Computing* C Heitmeyer and D Mandrioli (eds), John Wiley & Sons Ltd, 1996

[5] David Harel and Ammon Naamad, The STATEMATE Semantics of Statecharts ACM Trans Soft Eng Method. Oct 1996