ECA규칙을 이용한 동적 확장가능 로봇 아키텍처

°우희정[1] 정우성[1] 이은주[2] 이종석[3] 우치수[1]
[1]서울대학교 컴퓨터공학부, [2]경북대학교 컴퓨터공학과, [3]우석대학교 컴퓨터교육과
{hjwoo, wsjung, wuchisu}@selab.snu.ac.kr, ejlee@knu.ac.kr, jong1007@core.woosuk.ac.kr

A Dynamic Extendable Robot Architecture using ECA rules

°Heejung Woo[1] Woosung Jung[1] Eunjoo Lee[2] Jongsuk Lee[3] Chisu Wu[1]
[1]School of Computer Science and Engineering, Seoul National University,
[2]Dept. of Computer Engineering, Kyungpook University, [3]Dept. of Computer Education, Woosuk University

요 약

로봇은 특수 분야에서 일반 산업 및 가정에 이르기까지 다양한 경로를 통해 사람에게 유용한 서비스를 제공해왔다. 하지만 로봇의 아키텍처 및 인터페이스의 비표준화로 인해 호환성 및 생산성의 문제가 야기되었다. 본 논문에서는 ECA규칙을 이용하여 컴포넌트의 동적 추가 및 확장이 가능한 로봇 아키텍처를 제안한다. 센서와 동작기 부분의 인터페이스를 표준화하였으며 ECA규칙 적용을 위한 DB스키마 및 규칙 정보 관리를 위한 테이블 구조를 제시하였다. ECA규칙은 구조가 단순하여 기계학습의 응용에도 적합하다. 제안하는 아키텍처는 컴포넌트 조합이 바뀔 때마다 동적으로 새로운 규칙을 적용할 수 있고 공개된 컴포넌트가 가진 어휘를 이용하여 확장된 규칙을 정의할 수 있다.

1. 서 론

오늘날 로봇은 우주, 국방 등과 같은 특수 분야에서 일반 산업 및 가정에 이르기까지 다양한 목적 및 방식으로 사람들에게 혜택을 제공하고 있다. 그러나 로봇에 적용하는 기술, 기능들이 표준화되어 있지 않기 때문에 통합 및 개발에 있어 많은 어려움을 겪는다. 만약, 확장 가능하고 표준화된 로봇 시스템의 아키텍처를 공개하여 사용할 수만 있다면 다양한 로봇에서 호환이 가능한 컴포넌트들의 생산이 쉬워질 것이다. 즉, 로봇 시스템의 개발과 통합 비용을 줄이기 위해 모듈화와 표준화가 필요한 것이다[1]. 하드웨어에 의존적인 계층에서는 플랫폼과 운영체제, 통신 방법들의 이질성으로 인해 구현상의 표준화가 어려우므로, 소프트웨어 아키텍처 계층에서의 표준을 제안한다.

논문의 구성은 다음과 같다. 2장에서는 OMG(Object Management Group)에서 진행 중인 로보틱 시스템에 관한 연구와 규칙(Rule) 기반 유비쿼터스 칩, ECA(Event-Condition-Action)규칙에 관해 기술하고, 3장에서 로봇 아키텍처 및 모델 스키마에 대한 설명을 한 후, 4장에서는 ECA규칙을 적용한 확장 시나리오를 제시하고 결론을 맺는다.

2. 관련 연구

2.1 OMG의 로보틱 시스템(Robotic System)

OMG는 로봇 모듈의 재사용과 상호 운용을 위해 객체계층(Object Layer)의 표준화를 제안하였으며, 이 레벨에서의 로봇 모듈을 "RTCs(Robot Technology Components)"라고 명명하였다[2]. RTCs는 센서, 컨트롤러, 모터 등과 같은 기능을 지원하는 소프트웨어 모듈로써 다음과 같은 특징을 가진다.

- 각각의 RTC는 고유한 기능을 가짐
- RTC간 데이터와 명령어를 교환
- RTCs의 조합으로 새로운 RTC의 구성이 가능함

RTC는 공동으로 이용할 수 있으며 유연한 성질을 가지고 모듈화 되어야 하고 RTC의 조합과 분리가 가능하도록 다른 RTCs의 변화와 무관하게 개발 되어야 한다. 더불어, RTC간 인터페이스와도 구분되도록 구현되어야 한다. 또한, 다양한 로봇 시스템에서 쉽게 재사용될 수 있도록 보편적으로 사용가능하게 설계 되어야 한다.

2.2 유비쿼터스 칩(Ubiquitous Chip)

유비쿼터스 칩은 ECA규칙을 기반으로 하여 이벤트(Event)에 의해 진행되는 입출력 제어 장치를 유비쿼터스 환경에 적용시킨 것으로, 주변 환경 안에서 자동적으로 정보를 교환하고 그에 대한 반응으로써 동작(Action)을 취한다. 이러한 환경에서, 컴퓨터는 낮은 전력장치와 작은 용량의 메모리를 사용하면서도 자신의 기능을 동적으로 변화시킬 수 있어야한다. 만약 이러한 시스템의 구현에 C와 같은 프로그래밍 언어를 사용한다면 컴포넌트가 동작하는 동안에는 기능을 변경하기 위해 접근하는 것이 어렵다. 그래서 애플리케이션을 효과적으로 구성하기 위해 입출력에 대한 제어와 연결을 담당하는 부분과 ECA규칙을 기반으로 하여 접근하는 두 가지 특성을 구분하여 시스템을 구성한다[3]. 이렇게 하면 장치를 붙였다 떼었다 하는 식의 변경이 용이한 유연한 시스템을 만들기 쉬우며, 사람이 일반적으로 이해하는 실세계에서 인과적으로 일어나는 이벤트와 동작을 유비쿼터스 컴퓨터의 동작으로 기술하는 데에 규칙 기반 원리를 적용할 수 있다.

2.3 ECA규칙

이벤트가 발생하면 조건(Condition)이 이를 분류하고 동작을 실행하는 순으로 이루어진 동적인 규칙이다. 사용자가

본 연구는 한국과학재단 특정기초연구(R01-2006-000-11150-0) 지원으로 수행되었음.

이벤트와 조건에 해당하는 동작을 정의 할 수 있으며, 사용자 또는 규칙 결정자는 프로그래밍과 같은 낮은 레벨의 지식에 대해선 알 필요가 없다. 이벤트가 발생하는 것은 규칙의 원인이 되고 조건은 규칙을 발생시킬 때 필터의 역할을 한다. 즉, 조건에 참일 때만 동작이 실행될 수 있다[4]. ECA규칙은 동적으로 추가 및 변화가 가능하다[5].

3. 로봇 아키텍처

본 아키텍처는 진화 가능한 로봇 시스템의 모델로써 확장성, 유연성을 가지며 동적 적응이 가능한 구조이다. 그림1은 주요 인터페이스인 ISensor와 IActuator와 관련한 컴포넌트들 및 관계를 UML로 표현한 것이다.

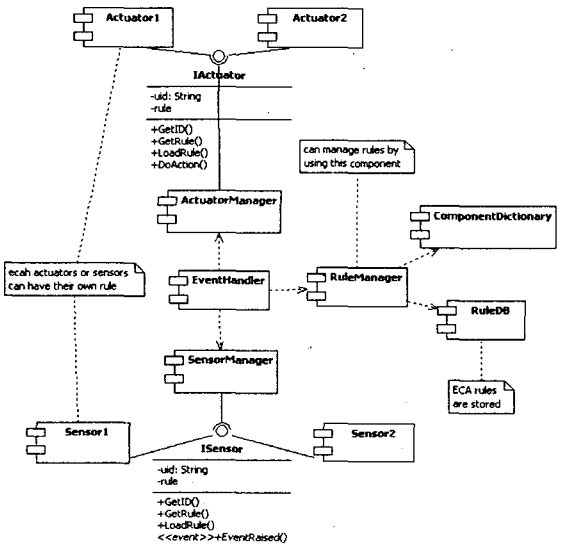

그림 1. 로봇 컴포넌트 다이어그램

센서(Sensor)와 동작기(Actuator)는 입력된 규칙에 따라 동작하게 되고 사용자가 이 규칙을 정의하고 추가시킬 수 있다. 초기에는 센서와 동작기에 해당하는 컴포넌트의 수가 적어 규칙을 나타내는 어휘력에 한계가 있으나, 컴포넌트에 사용되는 인터페이스를 단순 표준화시켜 공개하면 규칙에서 사용가능한 어휘가 증가하여 확장된 기능을 추가시키는 것이 가능하다. 이 경우 규칙에 사용하기 위한 이벤트와 동작은 알려진 센서와 동작기의 것이어야 한다. 규칙은 서로 충돌할 수 있기 때문에 각각의 이벤트와 동작에 대해서 우선순위를 정할 수 있어야 한다. 또한, 규칙은 시스템에서 추가되거나 센서나 동작기에 자체적으로 내장되어 있을 수 있다. 센서나 동작기에 규칙을 포함시키는 이유는 동적으로 시스템에 컴포넌트가 추가될 때 시스템의 구성요소가 변경되기 때문에 이에 따르는 새로운 규칙을 자동적으로 알 수 있어야 하기 때문이다.

그림2는 ECA규칙을 따르는 로봇 아키텍처의 동적 확장에 대한 시뮬레이션을 위해 다루게 되는 정보들 및 관계를 나타낸 것이다. 프로젝트가 최상위 테이블이며 복수개의 시스템을 포함할 수 있다. 그리고 시스템은 이미 정의되어 있는 센서 및 동작기의 인스턴스를 이용하여 구성됨을 알 수 있다.

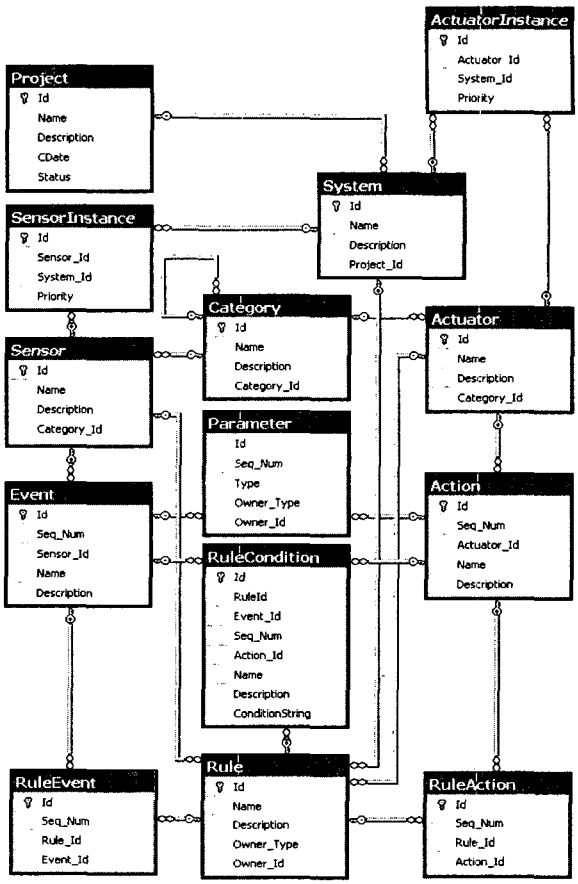

그림 2. ECA규칙 적용을 위한 DB스키마

센서와 동작기는 Id값으로 고유하게 구분되며, 각각 복수개의 이벤트와 동작이 정의된다. 해당 컴포넌트 내에서의 순서를 Seq_Num으로 구분하게 된다. 또한 각각의 이벤트와 동작과정에서 추가정보를 전달하기 위한 파라미터도 가지고 있다.

아직 ECA 컴파일러를 구현하진 않았으나, ECA 정보를 위한 DB스키마를 정의하여 모델을 이용한 시뮬레이션을 해본 결과 원하는 시나리오의 규칙을 따라 동작함을 확인할 수 있었다. 실제로는 이벤트에서 발생한 상태 값이나 파라미터를 받아서 이에 따르는 복잡한 조건에 따라 동작을 실행할 수 있어야 한다. 동작을 실행할 때도 파라미터 값을 통해 동작시키는 것이 보다 현실적이지만 본 연구에서는 모델의 단순화를 위해 이 부분이 제외 되었다.

4. ECA규칙 적용 예

규칙 정보는 다음과 같은 테이블로 이루어져 있으며 우선순위 순으로 정렬되게 된다. 새로운 센서나 동작기가 추가되거나 삭제될 때마다 규칙 테이블의 갱신이 이루어진다. 0번은 시스템 본체의 이벤트를 의미하며 동작의 경우도 마찬가지다.

표 1. 규칙 테이블

Sensor_Id	Event_Id	Condition	Actuator_Id	Action_Id
1	2	null	2	2
1	1	Params(1)>10	1	1
0	1	Params(1)<50	0	1
...	...	...	...	...

EventRaised(Sensor_Id, Event_Id, Params)는 콜백 함수로써 이벤트가 발생할 때 호출되며, 이때 센서의 고유번호와 이벤트 번호, 파라미터 값들이 전달된다. EventRaised 함수 안에서 Sensor_Id와 Event_Id값을 이용하여 검사해야할 Condition을 얻게 되고, 조건을 만족할 경우는 Actuator_Id와 Action_Id를 이용하여 동작기가 특정 동작을 수행할 수 있도록 한다. ECA규칙을 이용함으로써 센서와 동작기의 인터페이스를 최대한 단순화시킬 수 있다. 이는 자원을 효율적으로 이용해야 하는 임베디드 시스템의 특성상 중요한 점이다[6].

실제로는 Action_Id에 종속된 복수개의 RuleAction이 존재하여 DoAction(Actuator_Id, Action_Id)함수를 통해 여러 동작을 순차적으로 수행하게 할 수 있다.

표2는 센서에서 이벤트가 발생했을 때 규칙에서 필요한 조건을 만족시키는 동작을 수행하는 과정을 의사코드로 표현한 것이다.

표 2. 의사코드로 표현한 동작 과정

$$
\begin{aligned}
&Let\ s \in Sensor_{Id},\ e \in Event_{Id},\ c \in Condition,\\
&\quad a_1 \in Actuator_{Id},\ a2 \in Action_{Id},\ p \in Params,\\
&\quad < s, e, c, a_1, a_2 > \in Rule\\
&When\ \underline{EventRaised(s', e', p')}\\
&\quad for\ each\ < s = s', e = e', c = *, a_1 = *, a_2 = * > \in Rule\\
&\quad\quad If\ Satisfied(c, p')\ then\ \underline{DoAction(a_1, a_2)}
\end{aligned}
$$

그림3은 새로운 컴포넌트가 추가되어 로봇의 반응이 바뀌는 경우를 보여준다. 모델에서는 이벤트와 동작에 대한 파라미터 정보를 고려하였으나, 설명의 단순화를 위해 조건은 생략하고 특정 이벤트에 따른 동작을 실시하게 되는 경우로 한정한다. 특정 자극에 대한 규칙이 없었던 경우, 동작하지 않다가 새로운 동작기가 추가되면서 새로운 규칙이 적용되어 해당 자극에 대한 반응이 일어나는 과정을 나타낸 것이다.

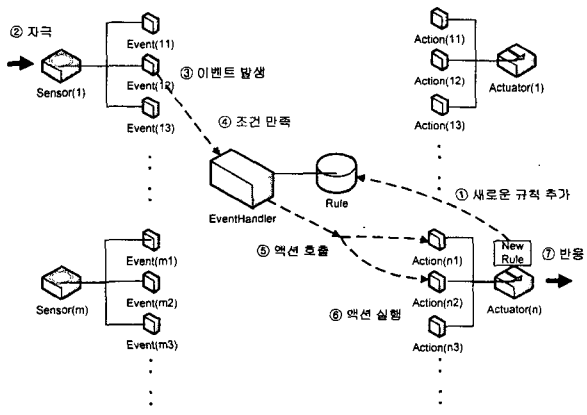

그림 3. ECA 규칙에 따른 동작 예

5. 결론 및 향후연구

본 연구는 컴포넌트의 인터페이스를 단순화시키고 이를 표준화하여 동적으로 확장 가능한 로봇 아키텍처를 제시하는데 그 목적이 있기 때문에 효율적인 동작 계획을 세우는 방법에 대해서는 다루지 않았다. 하지만 이처럼 자극에 반응하는 단순한 로봇의 경우는 ECA규칙을 이용함으로써 효율적인 방법으로 동적 확장이 가능한 시스템을 구축할 수 있다.

연구 수행 중 트랜잭션 문제나 보안 문제와 관련하여, 암호화된 키를 이용하여 컴포넌트의 결합을 제한하고, 우선순위를 제한할 수 있도록 할 필요가 있음을 확인하였다. 또한, 동작기가 순차적인 동작을 수행할 때 여러 이벤트들이 발생할 경우 이를 중재하여 처리할 수 있는 보다 효율적인 알고리즘이 필요한데, 현재로서는 동작이 충돌하는 경우 우선순위에 따라 인터럽트를 걸어서 처리할 수밖에 없다. 현재는 센서와 동작기 자체에 우선순위를 두고, 최종 ECA규칙 집합을 정렬하는 것으로만 처리하고 있다. 이는 보안과도 맞물려 있는 부분으로 좀 더 정밀한 우선순위의 조작에 대한 연구가 필요하다. 동일한 컴포넌트의 버전 구분에 따른 처리도 생략하였다.

하지만 ECA규칙은 구조가 단순하여 신경망 네트웍 등을 이용하여 기계학습을 시키는데도 적합하다. 다양하고 실제적인 시나리오들을 적용해 봄으로써 모델을 일반화시키고 우선순위나 보안과 관련한 부분에 대한 연구를 병행해야할 필요가 있다.

6. 참고문헌

[1] OMG, "Robotic Systems : Request For Information," *OMG*, June 2005.

[2] OMG, "Request For Proposal : Robot Technology Components(RTCs)," *OMG*, November 2005.

[3] T. Terada, M. Tsukamoto, K. Hayakawa, T. Yoshihisa, Y. Kishino, A. Kashitani and S. Nishio, "Ubiquitous Chip: A Rule-Based I/O Control Device for Ubiquitous Computing," *Lecture Notes in Computer Science*, no. 3001, pp. 238-253, 2004.

[4] Y. Jin, "A Design Pattern for Active Rule-Based System Architectures," *Proceedings of the IEEE International Conference on Information Reuse and Integration*, 2005.

[5] H. Ma, H. Johansson, K. Orsborn, "Distribution and synchronisation of engineering information using active database technology," *Advances in Engineering Software*, vol.36, no.11/12, pp.720-728, 2005.

[6] L. Sikun, X. Zhihui, L. Tiejun, "Distributed cooperative design method and environment for embedded system," *Proceedings of the IEEE Ninth International Conference on Computer Supported Cooperative Work in Design*, vol.2, pp.956-960, 2005.