WS-ECA 프레임워크 모델링

이강찬* · 이원석* · 이승윤* · 신동민**

*한국전자통신연구원, **한양대학교

Model of WS-ECA Framework

Kangchan Lee* · Wonsuk Lee* · Jonghong Jeon* · Seungyun Lee* · Dongmin Shin**

*Eletronics and Telecommunications Research Institute, **Hanyang Universoty

E-mail : {chan, wslee, syl}@etri.re.kr* · dmshin@hanyang.ac.kr

ABSTRACT

Finite automata는 실제 복잡한 컴퓨터의 연산을 수학적으로 모델링하는 computational 모델 중 하나로써 작은 메모리를 가지는 컴퓨터를 모델링하기에 적합하다. 이는 유비쿼터스 환경에서 실제로 작은 메모리를 가지는 장치가 ECA 룰을 작동시키는 디바이스 로직(디바이스 logic)을 효과적으로 운영할 수 있는지 여부를 검증하는 도구로 사용될 수 있다. 본 논문은 웹서비스 기반의 이벤트 처리 언어를 CFA로 모델링하고 알고리즘을 개발하였다.

키워드

Web Services, ECA rule, ubiquitous 디바이스, CFA

Ⅰ. 서론

최근 컴퓨터들 간의 메시지를 아무 제약 없이 주고 받을 수 있게 하기 위해 활발한 연구가 진행되고 있는 기술 중 하나가 웹 서비스다. 웹 서비스는 SOAP을 통해 어떠한 플랫폼이나 구현언어에 구애 받지 않고 기기간의 상호 운용을 가능하게 하며, WSDL과 UDDI를 통해 기기가 어떠한 서비스를 제공하는 지를 등록하고 검색할 수 있다. 그러나 웹 서비스를 그대로 유비쿼터스 환경에 적용하기에는 몇 가지 문제점들이 있다. 본래 웹 서비스는 사용자가 필요로 하는 서비스를 발견하기 위해 UDDI를 이용하고 있는데 이는 비즈니스를 목적으로 만들어졌기 때문에 좀더 광범위한 목적을 가지는 유비쿼터스 환경에서는 새로운 기술이 필요하다. 또한 웹 서비스의 상태가 계속 변화하고 있음에도 불구하고 웹 서비스 배포 시 저장된 UDDI는 이 변화를 감지할 수 없기 때문에 사용자는 UDDI에서 원하는 웹 서비스를 찾더라도 올바르게 작동하고 있는지 알 수 없다.

본 연구에서는 기존의 웹 서비스의 구조에 광역 규clr 처리기(global rule manager), 이벤트 소스(event source), 디바이스(디바이스)를 결합한 WS-ECA Framework을 통해 유비쿼터스 환경에 적용할 때 UDDI가 가진 문제를 해결하고자 한다. UDDI 대신 추가된 세가지 구성요소는 각 기기가 가지고 있는 서비스의 상태 변화를 감지하고, 그것에 대한 정보를 SOAP을 통해 다른 기기에게 전달함으로써 상호간에 서비스 상태를 공유할 수 있도록 한다.

ECA 기반의 룰 기술 언어는 웹서비스들이 주고받는 이벤트 알림 메시지를 기반으로 하고 있기 때문에 이벤트 알림 메시지에 효율적으로 접근하고, 활용할 수 있는 방법들을 제공한다.

따라서 본 연구에서는 중앙 집중적이고 정적인 정보만 다룰 수 있는 UDDI와 WSDL의 한계를 극복하기 위해선 서비스의 발견이 사용자의 위치, 환경, 필요에 따라 동적으로 이루어져야 하며, 서비스가 자신의 정보, 상태 등을 사용자에게 스스로 알릴 수 있어야 한다. 또한 사용자가 서비스를 쉽게 호출해서 사용할 수 있어야 하고, 사용자가 원하는 방식으로 가용한 서비스를 조합하여 사용할 수 있어야 한다. 우리는 이러한 조건들을 만족시키기 위해 ECA 기반의 룰 기술 언어를 정의하고 이것을 바탕으로 보다 효율적으로 웹서비스들

을 활용할 수 있는 방법을 제안하고자 한다. 이를 위하여 본 논문에서는 WS-ECA를 통하여 웹서비스 기반의 이벤트 처리 프레임워크를 제시하고, 모델링 라며, CFA로 알고리즘을 개발하였다.

II. WS-ECA 모델링의 특징

ECA 프레임워크는 여러 개의 디바이스로 구성이 될 수 있으며 여러 개의 디바이스들은 상호작용을 위해 서로의 상태에 영향을 미치게 된다. 그림 1에서 보는 바와 같이 어떤 디바이스가 이벤트를 받게 되면 저장되어 있는 ECA 룰을 동작시키게 된다. 이때 조건이 참이면 디바이스는 다양한 액션을 수행하게 된다. 예를 들면, 디바이스는 다른 디바이스에게 이벤트를 보내거나 이벤트를 자기 자신에게 전달하기도 하며 룰을 등록한 사용자에게 웹서비스를 제공하기도 한다. 이러한 액션들은 룰에 기술된 상태에 따라 하나씩 수행될 수도 있고 동시에 여러 개씩 수행될 수도 있다.

한 디바이스가 다른 디바이스에게 이벤트를 보내게 될 경우 이벤트는 다른 디바이스에 등록된 ECA 룰을 동작시키게 되고 조건이 만족되면 그에 맞는 액션을 수행하게 된다. 디바이스가 자신에게 이벤트를 발생시키는 경우에는 발생된 이벤트가 디바이스 내에 존재하는 또 다른 ECA 룰을 동작시키게 된다. 이렇듯 발생된 이벤트는 디바이스를 동작시키는 것이 아니라 디바이스 내에 존재하는 ECA 룰들을 동작시키게 되며 발생된 이벤트 또한 ECA 룰에 의해 생성되었다고 볼 수 있다. 즉, 한 ECA 룰에 의해 외부로 나가거나 혹은 내부로 전달되는 이벤트들은 모두 다른 ECA 룰들에 영향을 미치게 된다.

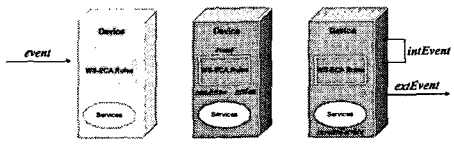


그림1. WS-ECA 프레임워크

이벤트를 받은 ECA 룰은 WS-ECA 프레임워크에서 정의된 디바이스의 논리(logic)에 따라 동작하고 액션을 수행하게 된다. 일반적으로 한 디바이스에는 여러 가지 ECA 룰들이 존재할 수 있으며 각각의 룰들은 서로 다른 종류의 이벤트, 조건, 액션이 정의되어 있다. 이러한 룰들이 오류가 없이 동작하고 액션을 수행하려면 앞서 정의된 디바이스의 동작 logic에 대한 검증이 필요하다.

그래서 본 연구에서는 WS-ECA 프레임워크 기반의 유비쿼터스 환경하에서 디바이스의 동작에 대한 로직을 모델링하고 검증하기 위해 finite state machine을 응용하고자 한다. 모델링의 중점은 하나의 ECA 룰이 디바이스의 동작 로직에 의해 운영되는 생명주기로 룰이 생성, 변경, 삭제되는 것과 더불어 이벤트에 의해 동작하고 액션을 수행하는 과정을 묘사할 것이다.

III. ECA를 위한 CFA

3.1 ECA 프레임워크 CFA(Communicating Finite Automata)

CFA를 구성하기 위해서는 우선 각각의 DFA를 먼저 구성해야 한다. 본 연구에서는 ECA가 생성되고 디바이스에 등록되어 실행되는 ECA의 생명주기를 기반으로 ECA 룰에 대한 DFA를 정의한다. ECA 룰의 DFA는 요청자에 의해 정의된 ECA 룰에 의해 받아야 할 input string이 정해진다. 이러한 것을 고려하여 다음과 같이 9개의 tuple을 가진 DFA를 구성할 수 있다. CFA를 구성하기 위해서는 우선 각각의 DFA를 먼저 구성해야 한다. 본 연구에서는 ECA가 생성되고 디바이스에 등록되어 실행되는 ECA의 생명주기를 기반으로 ECA 룰에 대한 DFA를 정의한다. ECA 룰의 DFA는 requester에 의해 정의된 ECA 룰에 의해 받아야 할 input string이 정해진다. 이러한 것을 고려하여 다음과 같이 9개의 tuple을 가진 DFA를 구성할 수 있다.

$$ECA_j = < Q_j, \Sigma_j, C_j, A_j, \delta_j, \gamma_j, \lambda_j, F_j, q_0 >$$

$Q_j = \{q_0, q_{\text{generate}}, q_{\text{operate}}, q_{\text{trigger}}, q_{\text{activate}}, q_{\text{mark}}, q_{\text{execute}}, q_{\text{end}}\}$: the state set of ECA rule

$\Sigma_j = \Sigma_{\text{ctrl}} \cup \Sigma_{\text{event}} \cup \Sigma_{\text{tf}} \cup \Sigma_{\text{m}} \cup \Sigma_{\text{r}} \cup \Sigma_{\text{action}}$: the set of input alphabets

$\Sigma_{\text{ctrl}} = \{\text{eca, sc, register, del}\}$,

$\Sigma_{\text{event}} = \{e_{\text{int}}, e_{\text{temp}}, e_{\text{ext}}\}$, $\Sigma_{\text{tf}} = \{\text{true, false}\}$,

$\Sigma_m = \{\text{mark, nomark}\}$, $\Sigma_r = \{\text{resolve, notresolve}\}$,

$\Sigma_{\text{action}} = \{a_{\text{assign}}, a_{\text{delay}}, a_{\text{invoke}}, a_{\text{passevent}}, a_{\text{int}}, a_{\text{ext}}\}$

C_j : the set of the conditions in ECA rule

A_j : the set of the actions in ECA rule $(A_j = \Sigma_{\text{action}})$

$\delta_j : Q_j \times \Sigma_j \rightarrow Q_j$: state transition function

$\gamma_j : Q_j \times \Sigma_j \rightarrow C_j$: condition generating function

$\lambda_j : Q_j \times \Sigma_j \rightarrow A_j$: action generating function

$F_j = \{q_{end}\}$: the set of accept state, q_0: initial state

$q_{trigger}$는 임의의 이벤트에 의해 ECA 룰이 trigger 된 상태를 나타낸다. 이 상태로 전이하려면 alphabet set Σ_j의 부분집합인 Σ_{event}에 있는 symbol 중 적어도 하나 이상을 입력 받아야 한다. $q_{activate}$는 조건이 참인 경우에 ECA 룰의 action을 취할 수 있는 준비 상태를 나타낸다. q_{mark}는 ECA 룰의 action을 수행할 때 dynamic conflict가 발생할 가능성을 가지고 있는 상태를 나타낸다. $q_{execute}$는 ECA 룰에 dynamic conflict가 발생할 가능성이 없다고 인지되거나 dynamic conflict가 해결되었다고 인지되어 action을 수행할 수 있는 상태를 말한다. 상태 $q_{execute}$에서 상태 q_{end}로 전이할 때는 alphabet set Σ_j의 부분집합인 Σ_{action}에 있는 symbol 중 적어도 하나 이상을 입력 받아야 한다.
이러한 ECA의 DFA는 ECA 프레임워크에서 여러 개 존재하며 이들은 서로의 상태에 영향을 미치게 된다.

3.2 전이 다이어그램과 정규 표현

조건은 웹서비스가 특정 작업을 수행하기 전에 그 작업을 수행해야 하는지 의사 결정을 할 수 있도록 하며, XPath의 boolean expression으로 표현된다.

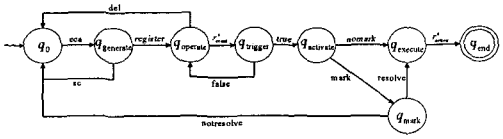
상태 $q_{operate}$에서 상태 $q_{trigger}$로 전이하기 위해서는 Σ_{event}에 있는 symbol중 적어도 하나 이상을 입력 받아야 한다. 이를 regular expression r^c_{event}로 표현하면 다음과 같다.

$$r^c_{event} = (e_{int} \cup e_{temp} \cup e_{ext})(e_{int} \cup e_{temp} \cup e_{ext})^* = \{x_1 x_2 \ldots x_k \mid k \geq 1, x_i \in \Sigma_{event}\}$$

상태 $q_{execute}$에서 상태 q_{end}로 전이하는 경우에도 앞의 경우와 유사하게 Σ_{action}에 있는 symbol들 중 적어도 하나 이상을 입력 받아야만 한다. 이를 regular expression r^c_{action}으로 표현하면 다음과 같다.

$$r_{action} = (a_{assign} \cup a_{delay} \cup a_{invoke} \cup a_{passevent} \cup a_{int} \cup a_{ext})$$
$$r^c_{action} = r_{action} r^*_{action} = \{x_1 x_1 \ldots x_1 \mid k \geq 1, x_i \in \Sigma_{action}\}$$

이러한 regular expression을 바탕으로 ECA를 다음과 같은diagram으로 나타낼 수 있다.



L(ECA)을 DFA가 인지하는 regular language라고 하자. 이를 표현한 regular expression은 다음과 같이 구할 수 있다.

Regular Expressions for ECA DFA

$((<eca><register><del>) \cup <sc> \cup (<register>r^c_{event}<true><mark><notresolve>))^*$
$<eca><register>(r^c_{event}<false>)^* r^c_{event}<true>(<nomark>r^c_{action}) \cup (<mark><resolve>r^c_{action})$

3.3 알고리즘

앞에서 모델링한 CFA를 이용하여 ECA 룰이 accept state에 도달하기까지의 알고리즘에 대해 설명한다.

$s = q_0$ // set initial state
$\eta = \text{eca}$
$s = \delta(s,\eta)$ // $s = q_{\text{generate}}$
if ($\eta = \text{registe}$)
  $s = \delta(s,\eta)$ // $s = q_{\text{operate}}$
  if $\eta = \text{e}$ // $\text{e} \in r^c_{event}$
    $s = \delta(s,\eta)$ // $s = q_{\text{trigger}}$
    $c = \gamma(s,\eta)$// generate condition from ECA rule
    if ($c = \text{true}$)
      $\eta = \text{true}$
      $s_0 = s$
      $s = \delta(s,\eta)$ // $s = q_{\text{activate}}$
      if ($\eta = \text{mark}$) // if ECA framework receive symbol "mark"
        $s = \delta(s,\eta)$ // $s = q_{\text{mark}}$
        if ($\eta = \text{resolve}$) // if ECA framework receive symbol "resolve"
          $s = \delta(s,\eta)$ // $s = q_{\text{execute}}$
          $a = \lambda(s,\eta)$// generate action
          $\eta = \text{a}$ // $\text{a} \in \Sigma_{\text{action}}$
          $s = \delta(s,\eta)$ // $s = q_{\text{end}}$
          stop
        else // $\eta = \text{notresolve}$
          $s = \delta(s,\eta)$ // $s = q_0$

stop

endif

else// $\eta = \text{nomark}$

$s = \delta(s,\eta)$// $s = q_{\text{execute}}$

$a = \lambda(s,\eta)$// generate action

$\eta = \text{a}$// $\text{a} \in \Sigma_{\text{action}}$

$s = \delta(s,\eta)$// $s = q_{\text{end}}$

stop

endif

else// $\text{c} = \text{false}$, $s = q_{\text{trigger}}$

$\eta = \text{false}$

$s = \delta(s,\eta)$// $s = q_{\text{operate}}$

stop

endif

else // $\eta = \text{del}$, $s = q_{\text{operate}}$

$s = \delta(s,\eta)$// $s = q_0$

endif

else // $\eta = \text{sc}$, $s = q_{\text{generate}}$

$s = \delta(s,\eta)$// $s = q_0$

endif

본 알고리듬은 WS-ECA 프레임워크에서 ECA 룰을 동작시키는 디바이스의 logic을 ECA 룰의 lifecycle을 모델링한 communicating finite state machine에서의 동작 logic으로 표현한 것이다. 이것은 WS-ECA 프레임워크를 구현하는데 있어 이론적 접근 배경을 제공함과 동시에 실제 대규모 시스템 구현을 위한 확장형(scalable) 모형이 될 수 있다.

IV. 결론

WS-ECA 프레임워크는 인간의 생활 전반에 내재되어 주어진 상황에 따라 스스로 동작해야 하는 기기들의 운용 메커니즘과 운용에 필요한 정보를 공유할 수 있게 한다. 즉, WS-ECA 프레임워크는 기기의 성능에 거의 영향을 받지 않는 단순한 메커니즘을 제공하고 서로 다른 플랫폼을 가지는 기기간에도 상호 운용이 가능하게 한다. 이는 유비쿼터스 환경 구현을 보다 효과적이고 효율적으로 할 수 있게 한다.

이를 위하여 본 논문에서는 WS-ECA 프레임워크에 대하여 ECA 구성 요소간의 전이에 대해서 CFA로 모델링하고 이에 대한 알고리즘까지 제시하였다. 향후 연구로는 이에 대한 성능 검증과 함께 수리 모델을 제시, 그리고 구현을 통하여 WS-ECA가 실제로 디바이스에 적용하였을 웹서비스들의 이벤트를 처리하기 위해서는 하나의 이벤트를 수신할 때도 다양한 조건을 제시할 수 있도록 하여, 동일한 이벤트를 받았을 때에도 다양한 조건식에 따라 원하는 작업을 수행할 수 있도록 하는 유연성을 제공하여야 하는 WS-ECA를 위한 웹서비스 확장 구조와 ECA의 주요 컴포넌트에 대해서 살펴보았다. 향후, 본 연구는 구체적인 ECA 기반의 룰 기술 언어 정의와 함께 룰 시스템에 대한 구현이 수반되어야 한다.

또한, 룰들은 다양한 디바이스에 저장되면서, 또는 다중 사용자에 의하여 룰이 정의되면서 서로 상충되거나 사용자가 원하지 않는 결과를 가져올 수가 있다. 이를 방지하기 위하여 룰 충돌 정의 및 충돌 시 이를 해결하는 메카니즘의 개발이 추가로 필요하다.

References

[1]Papamarkos, G., A. Poulovassilis, and P.T. Wood, Event-condition-action rules on RDF metadata in P2P environments. Computer Networks, 2006. 50(10): pp. 1513-1532.

[2]Paton, N.W. and O. Diaz, Active database system. ACM Computing Survey, 1999. 31(1): pp. 63-103.

[3]S. Vinoski. Integration with Web Services. IEEE internet computing, 7(6): 75-77, 2003.

[4]A. Carter and M. Vukovic. A Framework For Ubiquitous Web Service Discovery. In Proc. of the 6th UbiComp, 2004.

[5]A. Sashima, N. Izumi, and K. Kurumatani. Location-Mediated Coordination of Web Services in Ubiquitous Computing, in Proc. of IEEE Int'l Conf. Web Services (ICWS'04), pages 109-114, 2004.

[6]N. Bassiliades, and I. Vlahavas. 디바이스: Compiling production rules into event-driven rules using complex events. Information and Software Technology, 39:331-342, 1997.

[7]K. Liu, L. Sun, A. Dix, and M. Narasipuram. Norm Based Agency for Designing Collaborative Systems. Information Systems Journal, 11(3): 229-247, 2001.

[8]S. Calo, and M. Sloman, Policy-Based Management of Net-works and Services. Journal of Network and Systems Management. 11(3):249-252, 2003.

[9]J. Lobo, R. Bhatia, and S. Nagvi. A Policy

Description Language. In Proc. of National Conference of the American Association for Artificial Intelligence, Orlando, FL, 1999.

[10]M. Cilia and A. Buchmann. An Active Functionality Service for E-Business Applications. ACM SIGMOD Record, 31(1): 24-30, 2002.