유비쿼터스 응용을 위한 Annotation 기반 프로그래밍 언어

송교선 김민영 조은선 이강우 김현

An annotation-based programming language for ubiquitous applications

Gyo-Sun Song, Min-Young Kim, Eun-Sun Cho,
Kang-Woo Lee, Hyun Kim

요 약

유비쿼터스 환경에서의 응용프로그램은 다양한 데이터들과 그들의 연관관계 및 행위의 조합을 다루어야하므로, 일반적인 프로그램에 비해 복잡한 데이터 모델과 계산 모델을 필요로 하게 된다. 본 논문에서는 유비쿼터스 응용을 작성하는데 적합한 새로운 프로그래밍 언어를 제시한다. 사용의 편의를 위해 잘 알려진 Java를 기반으로 하고 있고, 기존의 통합 개발 환경을 그대로 사용할 수 있도록 하기 위해 문법 확장이 아닌 특수 주석(annotation) 과 API를 지원하는 방식을 사용하고 있다.

1. 서론

'유비쿼터스 컴퓨팅'이란 모든 사물이 사용자 서비스를 위해 상호 연결되고 지능화된 공간을 구성하기 위한 컴퓨팅 환경이다. 또한 컴퓨터는 눈에 보이지 않게 물리적 공간에 내재되어 제공되며 사용자의 상황에 따라 다양한 서비스가 제공되어야한다[1]. 사람과 주변 환경, 모든 장치등이 그 서비스에 따라 유기적으로 밀접하게 연동하게 되므로, 가능한 서비스 수와 개입되는 개체들의 수 및 그들의 관계의 복잡도는 이전 컴퓨팅 기술에서 상상되지 못했을 만큼 크고 동적이라고 할 수 있다.

이에 따라 기존의 프로그래밍 언어에서 기반으로 하는 모델만으로는 유비쿼터스 응용을 작성하기에 어려움이 생긴다. 특히 유비쿼터스 응용 프로그래밍을 위한 언어는 환경의 성격상 다음과 같은 사항들을 고려해야 할 필요가 있다.

- 다양한 소스로부터 추출되는 데이터를 참조해야 한다
- 외부 환경 변화에 대해 즉각적인 대응이 이루어져야 한다.
- 상호 연동이 잦으므로 외부 서비스 객체를 호출이 수월해야한다.
- 다양한 상황에 대한 처리가 필요하다.
- 수행되는 환경에 대한 가정이 주어지기 전에도 프로그래밍이 가능해야한다.

본 논문에서는 이와 같은 필요성들을 고려한 새로운 유비쿼터스 응용 프로그래밍 언어를 제안한다. 사용의 편의를 위해 잘 알려진 Java를 기반으로 하고 있고, 기존의 개발 환경을 그대로 사용할 수 있도록 하기 위해 문법 확장이 아닌 주석(annotation) 방식을 사용하고 있다.

현재는 ETRI의 유비쿼터스 상황인식 시스템 미들웨어인 CAMUS (Context-Aware Middleware for URC Systems) 시스템에 탑재되도록 설계하였으며 따라서 CAMUS의 데이터 모델인 UDM(Ubiquitous Data Model) 에 기초하여 프로그래밍 할 수 있게 하고 있다. 그러나 사용된 메카니즘은 다른 시스템과 데이터 모델에 충분히 이식가능하다.

2. 유비쿼터스 데이터 모델 (UDM)

UDM에서는 모든 상황 정보를 노드와 노드 사이의 연관으로 표현한다. 노드는 가상공간에서 개체를 표현하며 사람, 장소, 작업, 서비스 등이 된다. 모든 노드는 각각 고유한 식별자와 타입을 갖는다. 또한 노드의 특정 타입으로 "valued" 노드가 있으며, 이는 임의의 Java 객체로부터 값을 얻어올 수 있는 노드이다. 연관은 노드 사이의 관계를 기술하며, 방향성을 갖는 화살표로 표현되며 그 의미를 표현하는 이름을 갖는다. 화살표가 시작되는 노드를 시작 노드(from node)로, 화살표가 끝나는 노드를 종료 노드(to node)라 부른다. 그림. 1은 UDM을 통한 모델링의 예를 보여준다.

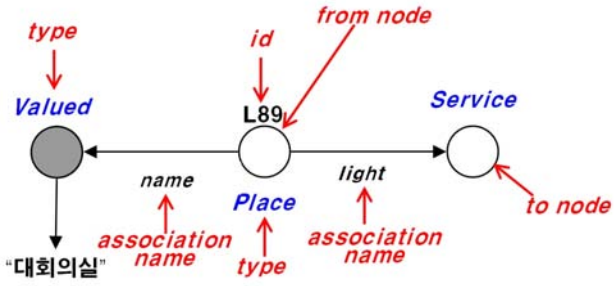

그림. 1 Universal Data Model

UDM으로 정의된 데이터는 별도의 파일에 저장되고 프로그래머들은 이것을 숙지하고 응용을 작성한다. 그 파일은 UDM의 각 노드 및 연관을 정의한 것으이다. 다음은 그 간단한 예의 일부이다.

```
UDM_model {
        user User {
                Range preferred_temp;
        };
        service Light {
                boolean power;
        };
        service AirConditioner {
                boolean power;
                int desired_temp;
        };
        sensor Temporature = int;
        place Place {
                multiple Light light;
                U_AirConditioner aircon;
                U_Temporature temp;
...
```

3. 제안하는 언어

3.1 경로

UDM 으로 표현된 데이터는 경로 표현식으로 나타내어질 수 있다. 가령 그림 1에서 현재 장소가 L89라면 '$place.name'은 "대회의실"을 나타낸다. 이러한 경로 표현식은 Java와 같은 객체지향 프로그래밍 언어나 XML 등에서는 익숙한 식이므로 프로그램에서 자연스럽게 이용될 수 있다.

제안하는 언어도 Java 내에 경로 표현식이 문자열로 전달되는 함수를 제공하고 있다. 이에 덧붙여, 일반 객체와 동일한 구조로 사용될 수 있도록 관련 API들을 지원하고 있다. 다음 예에서 Java 변수 $place가 나타내는 노드의 resident 에 해당하는 노드들은 한명 이상이라고 가정한다.

```
Iterator iter =
    $place.pathQuery("resident");
while (iter.hasNext()){
    UDM_Node node = (UDM_Node)iter.next();
}
```

UDM_Node는 하나의 UDM 노드를 나타내며, 이를 시작노드로 하여 종료 노드들로 나가는 여러가지 연관관계들은 내부적으로 List로 표현되어 순서가 부여된다. 동일한 이름의 다수의 연관관계가 서로다른 종료노드를 가지는 것도 가능하다.

경로를 따라가는 중간에 만나는 노드나 결과적으로 취해올 실제 데이터는 다양한 형태로 저장되게 된다. 제안하는 언어에서는 UDM과 경로라는 공통적인 모델에 부합되기만 하면 같은 인터페이스로 접근이 가능하도록 하기 위해 경로 인터프리터 인터페이스를 두고 개개의 자료 형식에 맞는 구현 모듈을 끼워넣도록 하고 있다. 현재는 XML형식의 데이터와 계층적 파일 시스템에 대해 처리하고 있다.

3.2 작업규칙

CAMUS 를 포함한 많은 유비쿼터스 시스템들은 응용프로그램이 주어진 상황 이해 및 그 상황에 대한 즉각적 반응을 위해 작업 규칙을 지원하고 있다[3,4].

제안하는 언어에서는 사용자가 보다 쉽게 작업규칙을 기술할 수 있도록 ECA (Event-Condition-Action) 규칙 표현을 기초로 작업 규칙을 구성할 수 있도록 하고 있다. 이를 위해 새로 Java 문법을 직

접 확장하기 보다는 특수 주석을 사용하여 규칙에 관한 부분을 표현할 수 있도록 하고 있다.

```
/**
@OnEvent.method name = "Light"
                location = "m_place"
                event = "UserEntered"
*/
public void Light()
{   Iterator iter =
        m_place.getPathQuery("light");
    while (iter.hasNext()){
        UDM_Node node =
            (UDM_Node)iter.next();
        if (!node.getValue("power"))
          node.call("turnOn");
 }
```

Light()은 해당 장소 (m_place)에 사용자가 입장하면 유발되는 이벤트에 반응하는 메소드이다. 이 예에서는 사람이 들어오면 불을 켜는 작업에 대한 정의를 보여준다.

발생되는 이벤트 (event)에 대한 기술은 특수 주석으로 처리되고 있으며, 이에 대한 대응행위 (action)에 해당하는 것은 Light() 메소드로 기술되고 있다. 현재 ECA의 조건 (condition) 부분은 메소드 내에서 검사하는 것으로 하고 별도의 특수 주석은 추가하지 않았으나, 수행 규칙을 선택하는데 적용되는 조건에 대해서는 확장할 계획 중에 있다.

주석에 대한 처리는 소스 내에 특수하게 주석 처리된 문장을 읽고 동작하는 Xdoclet[5]을 사용하고 있다. 즉, 소스 내에 간단한 주석(예에서는 @OnEvent.method 절) 을 작성하기만 하면 컴파일 시에 필요한 메타데이터들이 자동적으로 만들어지게 된다. 이러한 XDoclet은 편리함과 확장성을 가지고 있어서 많은 이들이 사용하고 많은 plug-in들이 만들어 지고 있다. 널리 사용되고 있는 통합 개발 환경인 이클립스(eclipse)용 플러그-인 (plug-in) 을 사용하면 쉽게 XDoclet의 동작을 확인할 수 있다는 잇점도 가진다.

본 논문에서 제안하는 언어 외에도 현재 CAMUS 시스템을 위한 또 다른 프로그래밍 언어로 PLUE(a Programming Language for Ubiquitous Environment) 가 있다[2,6]. PLUE는 Java 문법자체를 확장하는 접근방법을 취하고 있으며, 유비쿼터스 응용 프로그램을 위해 막강한 기능을 더 많이 표현할 수 있다. 앞서 Light()메소드와 연관된 규칙을 PLUE로 표현하면 다음과 같다. 일반 Java 문법에 이벤트 관련 규칙과 경로 표현식이 그대로 병합되어 있는 것을 볼 수 있다.

```
on ( m_place::UserEntered e ) {
    foreach ( light = m_place.light: !light.power )
    {
        light.power = true;
    }
}
```

제안하는 언어는 PLUE와 상호 보완적인 관계를 가진다. 즉, PLUE 는 풍부한 구조를 지원하고 사용자의 편의를 제공함으로써 더욱 복잡한 응용에 적합하다. 하지만, 이클립스등 기존의 통합 개발 환경 (IDE) 과 함께 사용하였을 때 오류 검출의 난해성이 내재되어 있으므로 해당 도구를 사용하는 프로그래머들에게는 실용성 면에서 본 논문에서 제안하는 방식이 더 편리할 것으로 본다.

4. 구조 (architecture)

그림 2와 같이 Xdoclet을 위한 특수한 주석문 (annotation)을 포함한 소스코드를 XDoclet 라이브러리를 포함하여 컴파일을 하면 Java의 바이트코드와 XDoclet에 의해서 생성된 XML 데이터가 결과로 나온다.

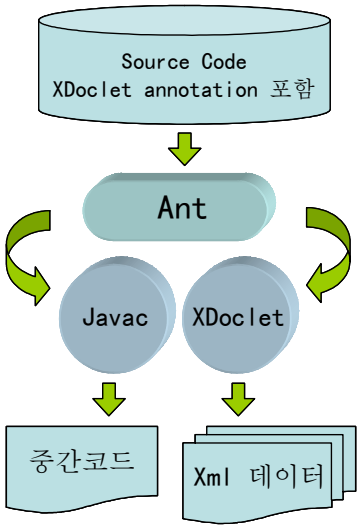

그림 2 컴파일시 구조

다음은 Light() 메소드에 대한 주석이 XDoclet 처리된 결과로 나온 XML 파일이다. 이벤트와 연관되는 메소드를 태그로 정의하여 항목으로 설정하고 있다.

```
<OnEventStatement>
   <Method name = "Light" location = "m_place">
      <Event content = "UserEntered">
      </Event>
   </Method>
</OnEventStatement>
```

앞서 나온 두 결과는 런타임에 실제 수행이 일어날 때 사용된다. 그림 3과 같이 이벤트 핸들러(Event Handler)에서 UserEntered 와 같은 이벤트를 감지하게 되면 컴파일 시 XDoclet에 의해 생성된 XML 데이터가 참조되어 대응 행위가 Light 메소드라는 것이 파악된다. 컴파일되어 준비된 Light 메소드는 이 때 동적으로 로딩 되어 실행된다. 따라서 주어진 이벤트에 대한 동적인 즉각적인 대응을 가능하게 한다.

5. 결론 및 토의

기존의 유사한 형태의 언어들은 더러 존재하였지만, 유비쿼터스 응용 프로그래밍을 위한 언어는 아직 거의 등장하지 않고 있다. u-시스템 설계에 대한 고려가 우선시 되어 정작 응용 프로그래밍 자체에 대한 기술적인 고찰이 미흡했던 것으로 보인다. agent 기반 유비쿼터스 시스템에서 간혹 사용한 규칙 스크립트 언어는 본격적인 응용 프로그래밍 작성하기에는 부족하다.

본 논문에서는 유비쿼터스 프로그래밍을 위한 프로그래밍 언어를 제안하였다. Java를 기반으로 하였고, XDoclet 을 사용하여 규칙을 특수주석으로 처리하여 상황에 대한 즉각적인 반응을 지원한다. 또한 경로 표현식 형태의 유비쿼터스 데이터 모델에 대해 Collection과 Iterator로 처리 할 수 있도록 API를 제공하고 있다. 이러한 접근 방법은 기존의 언어를 문법적으로 확장하는 방식에 비해 표현의 한계를 가지나 통합 개발 환경에서 자유로이 사용할 수 있다는 장점을 가진다.

현재는 ETRI의 유비쿼터스 서비스 미들웨어인 CAMUS 위에 탑재를 진행 중이나, 향후 다른 시스템에도 적용할 수 있다. 조건 (condition) 부분을 특수 주석처리 하는 것과 더 많은 표현식을 특수 주석으로 가능하게 하고 연결되는 메소드와의 정보 공유를 늘일 수 있도록 지원하는 것을 계획 중에 있다.

참고문헌

[1] Mark Weiser, "Some Computer Science Problems in Ubiquitous Computing," Communications of the ACM, July 1993.

[2] H. Kim, Y.J. Cho, S. R. Oh, "CAMUS: A middleware supporting context-aware services for network-based robots", 2004, *submitted for publication*

[3] M. Roman, C.Hess, R. Cerqueira, A. Ranganathan, RH Campbell, K. Nahrstedt. Gaia: A Middleware Infrastructure for Active Spaces. IEEE Pervasive Computing Vol. 1, No. 4, p. 74-83. October - December, 2002

[4] Kenichi Takahashi, Satoshi Amamiya, Tadashige Iwao, Guoqiang Zhong, M. Amamiya: An Agent-based Framework for Ubiquitous Systems, Proc. of the 2nd International Workshop on Challenges in Open Agent Cities, pp. 49-52, July. 2003

[5] Craig Walls and Norman Richards, XDoclet in Action, ISBN 1932394052, Manning Publications Co., 2003,

[6] Eun-Sun Cho, Kang-Woo Lee,"Security Checks in Programming Languages for Ubiquitous Environments", in Proc. of Workshop on Pervasive, Security, Privacy and Trust (PSPT) 2004, Aug., 2004, Boston, USA