Context Toolkit 에서의 동적 확장을 위한 Dynamic Widget Binder

황정섭, 류은석, 유혁
고려대학교 컴퓨터학과
e-mail : jshwang@os.korea.ac.kr

Dynamic Widget Binder for a Dynamic Extension in the Context Toolkit

Jeong-Seop Hwang, Eun-Seok Ryu, Hyuck Yoo
Dept. of Computer Science and Engineering, Korea University

요 약

유비쿼터스 환경에서 컨텍스트의 의미가 점차 중요해진다. 시간이 지날수록 더욱 더 많은 컨텍스트가 우리 주위를 장식해가고 많은 정보들을 제공해 줄 것이다. 이러한 상황에서는 새롭게 추가되는 컨텍스트를 유지하고 관리하는 기법들이 중요한 이슈로 자리잡게 된다. 하지만, 기존의 Context Toolkit 에서 이렇게 동적으로 변하는 컨텍스트들을 효율적으로 유지, 관리할 수 있는 방법이 없었다. 이에 따라, 우리는 새롭게 추가되는 컨텍스트들을 유지, 관리하고 동적으로 어플리케이션에게 바인딩 해 주어 미리 컨텍스트들에 대한 환경 설정을 해야 하는 부담을 줄일 수 있도록 하는 DWB (Dynamic Widget Binder)에 대한 개념을 소개한다.

어플리케이션이 어떠한 컨텍스트를 사용하려고 할 때, 어플리케이션은 DWB 에게 컨텍스트를 요청한다. DWB 에서는 요청한 컨텍스트가 존재하는지 확인한 이후, 컨텍스트가 존재한다면 요청한 어플리케이션으로 콜백을 걸어주고, 컨텍스트가 존재하지 않는다면, 시간이 지나 요청한 컨텍스트가 추가 될 때, 어플리케이션으로 관련 내용을 통보해 준다. 새로운 컨텍스트가 추가되었을 때, 컨텍스트는 DWB 에 자신을 등록하여 DWB 가 자신을 요청하는 어플리케이션과 바인딩 시켜줄 수 있도록 한다. 이처럼 컨텍스트를 유지, 관리하는 DWB 의 개념을 추가하여 어플리케이션은 동적으로 추가, 삭제되는 다양한 종류의 컨텍스트를 쉽게 이용할 수 있다.

우리의 연구는 DWB 의 개념 정립과 더불어 Context Toolkit 에 관련 모듈을 추가하여 구현함을 목표로 한다.

1. 서론

유비쿼터스(Ubiquitous)라는 단어는 더 이상 우리에게 낯선 단어가 아니다. 더 이상 시간과 장소의 구애를 받지 않는 시대가 오고 있는 것이다. 이러한 상황에서 사용자의 주변 환경, 이를테면, 온도, 밝기, 습도와 같은 환경 요소들은 사용자가 다른 행동 패턴을 가지도록 한다. 따라서 이러한 환경 정보는 사용자의 패턴을 파악하고 적절한 서비스를 제공하는데 있어서 중요한 정보가 된다. 이렇게 환경 정보를 얻을 수 있는 Environment Sensing 의 중요성이 더욱 커진다.

주변의 다양한 환경 정보를 얻기 위해서 각 어플리케이션이 센싱 장치 하나하나에 직접 접근하고 정보를 얻어 오는 것은 불합리한 점이 많다. 개발자의 입장에서 하나의 어플리케이션을 개발하기 위해 어플리케이션 본래의 기능을 구현하는 것에 충실하지 못하고 센싱 장치에 접근하고 정보를 얻어 오는 작업에 많은 시간을 투자해야 한다. 이것은 결국 개발자로 하여금 시간과 인력의 낭비를 초래하게 한다. 따라서 환경 정보를 얻을 때에, 각 센싱 장치에 접근하여 정보

본 연구는 한국정보통신대학교 디지털미디어연구소의 정보통신연구개발사업의 연구비 지원에 의하여 수행되었음

를 얻어오는 미들웨어(middleware) 계층을 추가하고, 이들의 상위 어플리케이션이나 혹은 다른 미들웨어에게 필요한 정보를 제공해 줄 수 있다면, 어플리케이션을 개발하는 데 있어서 많은 수고를 덜 수 있다.

우리의 연구 목표는 이렇게 다양한 환경 정보를 어플리케이션이나 미들웨어 컴포넌트들이 쉽게 사용할 수 있도록 하기 위한 툴킷(toolkit)을 제공하는 것이다. 툴킷을 이용함으로써 어플리케이션이나 미들웨어 컴포넌트는 환경 장치에 대한 구체적인 사항들을 알 필요 없이 충분히 사용할 수 있게 된다. 또한 쉽게 하위의 환경 장치(context)를 추가하거나 제거할 수 있으며, 환경 장치의 위치에 구애됨이 없이 이용할 수 있도록 도와준다.

기존의 툴킷이 가지고 있는 제한적인 기능을 극복하고 동적으로 widget 을 추가, 제거할 수 있는 기능을 추가하여 어플리케이션에게 좀 더 유연한 서비스를 제공해 주는 미들웨어로 확장하는 것이 우리의 목표이다.

2. 관련 연구

컨텍스트(Context)란 어떠한 하나의 엔티티(Entity)의 상황을 특징짓는 데 사용될 수 있는 임의의 정보를 말한다. 가령, 자동 습도 조절 장치가 있다면 현재의 습도에 의해서 습도 조절 장치가 동작이 되어야 할지, 그렇지 않을지를 결정하게 되므로 습도는 하나의 컨텍스트가 된다. 이러한 정의에 따르면 사람, 장소, 물리적 오브젝트 등 변화를 줄 수 있는 모든 것들을 포함하게 된다.

이러한 컨텍스트는 다음과 같은 특징을 지닌다.

▪ 컨텍스트의 소스(source)는 다양하며, 여러 위치에 분산되어 있다.

▪ 컨텍스트는 추가적인 추상화를 필요로 한다. 예를 들면, GPS 는 위도, 경도의 정보를 제공하지만 어플리케이션은 이러한 정보를 통해 사용자가 위치하고 있는 장소(예를 들면 도로의 이름)를 필요로 한다. 이를 위한 추상화를 필요로 한다.

▪ 정보를 얻는 컴포넌트(widget)는 어플리케이션에 종속적이지 않다. widget 은 때때로 동시에 동작하는 여러 어플리케이션에게 정보를 제공해야 하는데 이러한 상황에서는 widget 이 어플리케이션과는 독립적이어야 할 필요가 있다.

이러한 컨텍스트를 쉽게 이용하고 상위 어플리케이션에게 하위 컨텍스트 정보들을 제공하기 위한 미들웨어로 Context Toolkit[1,2]이 있다. 이것은 Georgia Institute of Technology 의 Day 에 의해 이루어진 프로젝트이며 하위 컨텍스트들에 대한 적절한 추상화(abstraction)을 통해서 심지어 컨텍스트가 로컬(local)에 있던 리모트(remote)있던지 상관없이 어플리케이션이 이용할 수 있는 방법을 제공한다.

Context Toolkit 은 크게 다음의 세 가지로 이루어져 있다.

▪ Widget

컨텍스트를 사용하는 어플리케이션이 그 하단의 세부적인 내용을 알 필요가 없도록 단일화된 인터페이스로써 컨텍스트를 이용할 수 있도록 하기 위한 개체이다. 보통 하나의 컨텍스트를 래핑(Wrapping) 하고 있다.

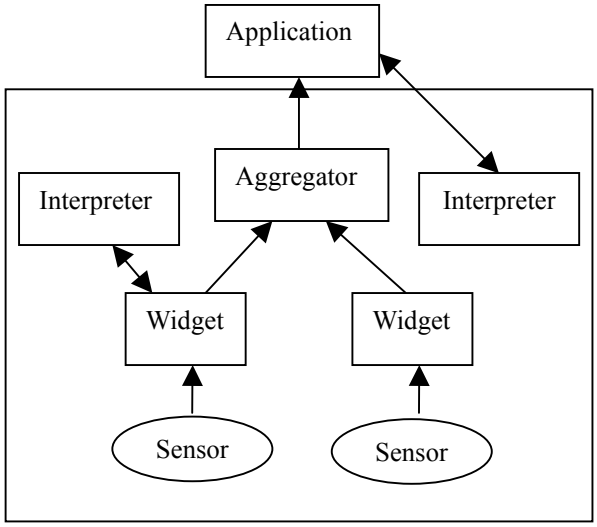

그림 1. Context Toolkit 의 아키텍처

▪ Aggregator

기본적인 widget 들과 응용 프로그램 사이의 통로(gateway) 역할을 함으로써 어플리케이션이 두 개 이상의 여러 widget 의 정보를 통합(integrate) 할 수 있도록 한다.

▪ Interpreter

하위 개념의 컨텍스트 정보들을 보다 상위 개념의 정보로 해석해 주는 개체이다. 예를 들어 사용자 아이디, 위치, 소리의 크기 등의 정보들을 통해 현재 방안에서 회의가 이루어지고 있음을 유추해 낼 수 있다. 이러한 일은 Interpreter 를 통해 이루어진다.

3. DWB (Dynamic Widget Binder)의 개요 및 설계

3-1 DWB 의 개념

본 논문에서 제시하는 DWB 는 주변의 여러 컨텍스트들에 대한 정보들을 저장하고 있는 일종의 데이터베이스이다. DWB 는 새로운 컨텍스트가 들어올 때 컨텍스트의 정보를 받아들여 그들을 유지, 관리하며, 어플리케이션의 요청에 따라 어플리케이션이 원하는 컨텍스트를 제공해준다[3,4].

어플리케이션 및 DWB 의 관계는 그림 2 와 같이 구성되어 있다. 어플리케이션의 입장에서 보았을 때, DWB 는 필요로 하는 컨텍스트를 제공해 주는 Lookup 서비스라고 볼 수 있다. 따라서, DWB 는 Discover 의 내부에 독립된 쓰레드로 동작하며 어플리케이션으로부터 들어오는 요청을 처리하는 역할을 한다.

또한 컨텍스트를 래핑(Wrapping) 하고 있는 Widget 은 DWB 가 어플리케이션에게 자신을 제공해 줄 수 있도록 미리 DWB 에 등록한다. 자신을 이용할 수 있도록 하기 위해서 widget 은 DWB 에게 자신의 충분한

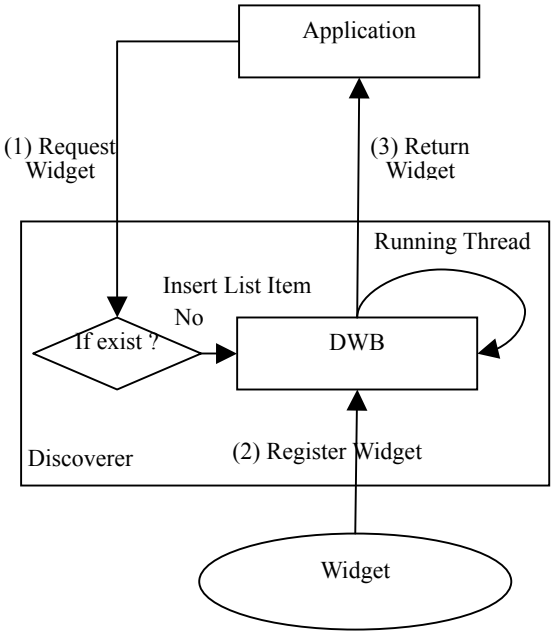

그림 2. 각 컴포넌트 간의 관계

정보를 제공해야 하며 여기에는 서비스의 이름, 타입 및 세부 파라미터 호출과 같은 충분한 API 정보가 포함되어야 한다.

3-2 DWB 의 세부 동작 과정

DWB 가 어떻게 동작하는지, 어플리케이션이 필요로 하는 widget 을 요청했을 때의 흐름은 다음과 같다.

1) 어플리케이션이 Discoverer 에게 자신이 필요로 하는 widget 을 요청한다.
2) Discoverer 는 어플리케이션의 요청을 받아들이고 DWB 내에 어플리케이션이 요청한 type 의 widget 이 존재하는지 확인한다.
3) 만약 DWB 내에 widget 이 존재하면, DWB 는 관련 widget 을 어플리케이션에게 전달한다.
4) 만약 DWB 내에 widget 이 존재하지 않는다면, DWB 는 요청했던 어플리케이션 및 필요로 하는 widget 의 정보를 저장해 둔다.

어플리케이션이 특정 widget 을 요청하고 해당 widget 이 DWB 에 등록이 되지 않았을 때, 어플리케이션의 요청 타입에 따라서 두 가지의 모드가 존재한다. widget 이 존재하지 않을 때, DWB 는 자신에게 존재하지 않는다는 사실만을 어플리케이션에게 통보할 수 있고, 추후에 어플리케이션이 원하는 widget 이 등록되었을 때, 이를 어플리케이션에게 연결해 줄 수도 있다. 이러한 모드를 어플리케이션이 widget 을 요청할 때 전송하도록 하여 모드에 따라서 DWB 의 행동을 결정짓도록 한다.

어플리케이션이 widget 을 요청하는 것과는 반대로 widget 이 새롭게 추가 되었을 때의 흐름은 다음과 같다.

1) widget 이 가동된다.
2) DWB 에 자신의 정보를 등록한다. 여기에는 해당 메시지 및 핸들러가 포함된다.
3) DWB 는 받아들인 정보를 List 로 유지한다.
4) DWB 는 대기 어플리케이션 리스트를 통해 해당 widget 을 요청한 어플리케이션이 존재하는지 확인하고 원하는 어플리케이션에게 해당 widget 을 바인딩 시켜준다.

3-3 DWB 의 프로토콜

DWB 의 메시지 프로토콜은 크게 보면 두 가지로 구성된다. 우선 어플리케이션이 자신이 필요로 하는 widget 을 얻기 위하여 Discoverer 에게 보내는 Request 메시지가 있고, widget 이 자신을 등록하기 위해서 DWB 에게 보내는 Registration 메시지가 존재한다. 이제 이러한 두 가지 메시지의 포맷 및 프로토콜에 대하여 알아보자.

Context Toolkit 은 다양한 종류의 센서들을 지원한다[5]. 이를 위해 우리는 미리 이러한 센서들을 분류하여 type 을 정의하고 각 type 별로 센서들을 정의하는 id 를 정의한다. 이렇게 정의되어 있는 센서들의 몇 가지 리스트는 다음과 같다.

Type	Type ID
Audio	0x03
Video	0x04
Location	0x05
Humidity	0x06
...	...

표 1. 정의되어 있는 센서 타입 리스트

이렇게 정의된 type 정보를 가지고 각 센서들을 구분하기 위한 세부 type 이 존재한다.

Sub Type	SubType ID
Sound Level Indicator	0x01
Voice Recognition	0x02
...	...

표 2. Audio 에 대한 세부 타입 리스트

표 1,2 에서 보는 바와 같이 센서들은 센서들의 카테고리에 해당하는 type 정보와 같은 type 정보 내에서 센서들을 구분하기 위한 sub type 정보를 가진다. 이러한 두 가지 정보를 조합함으로써 완성된 센서 ID 가 결정된다. 예를 들어 위의 표에서 Sound Level Indicator 는 Audio(0x03)에 속하고 Sub Type ID 가 0x01 이므로 전체 ID 는 0x0301 이 된다.

유비쿼터스 환경에서는 특정 종류의 센서가 반드시 하나만 있어야 하는 조건은 없다[5,6]. 즉, 같은 센서가 주위에 여러 개가 존재할 수 있다. 그러므로 위와 같은 센서 ID 만으로는 주위에 퍼져 있는 각 센서들을 정확히 가리킬 수 없다. 그러므로 센서에 대한 Type ID 외에도 각 센서마다 고유한 값을 가지는 Sensor ID 를 추가로 가져야 한다. 수 많은 센서들이

들어왔다 나가는 상황에서 우연히 같은 ID 를 가진 센서들이 만나는 일이 없도록 하기 위하여 SID (Sensor ID)는 고유한 값을 가져야만 한다.

어플리케이션이 Discoverer 에게 보내는 Request 메시지의 포맷은 다음과 같다.

Field	Meaning
SrcApp	자신의 어플리케이션에 대한 객체
WType	필요로 하는 widget 의 타입
Delay	자신이 기다리는 Delay
FWait	필요로 하는 widget 이 없을 경우 나중에라도 해당 widget 을 받기를 나타내는 플래그

표 3. Request 메시지의 각 필드 및 의미

어플리케이션은 widget 을 요청할 때, 만약 DWB 가 해당 widget 의 정보를 가지고 있지 않다면, 추후에라도 해당 widget 을 얻기를 원하는 지 그렇지 않은지를 메시지에 담아 보낼 수 있다. 이럴 때에 사용하는 필드가 FWait 필드이다. 이것이 참이면 어플리케이션은 widget 을 비동기적으로 기다린다.

widget 이 자신을 등록 할 때 DWB 로 전달하는 Registration 메시지 포맷은 다음과 같다.

Field	Meaning
WType	자신의 widget 타입
SID	widget 의 Unique Name
TExprie	widget 의 만료 시간
SrcWidget	자신을 가리키는 객체
Description	자세한 세부 정보를 담고 있는 URI

표 4. Registration 메시지의 각 필드 및 의미

widget 은 타입 정보 및 자신의 고유한 ID 를 가지며 만료 시간에 대한 정보를 가진다. 또한 자신의 세부 정보를 웹 등과 같은 곳에 올려놓고 해당 문서의 URI 를 전달함으로써 widget 에 대한 보다 자세한 정보를 얻을 수 있도록 한다.

4. 결론 및 추후 연구

이렇게 우리는 Context Toolkit 을 확장하여 동적인 특성을 갖는 widget 을 지원할 수 있도록 DWB 의 개념을 도입하고 메커니즘을 설계하였다. 이를 통해서 자유로이 들어왔다가 나가게 되는 widget 들의 환경을 미리 설정할 필요 없이 동적으로 어플리케이션이 이용할 수 있도록 지원하였다.

동적인 특성을 부여함으로써 유비쿼터스 환경에 보다 잘 적응하도록 할 수 있었고, 또한 추후의 개발 환경에서도 충분히 이용 가능하도록 서비스를 제공할 수 있다.

이러한 특성을 바탕으로 보다 신뢰성 있는 데이터를 어플리케이션이 얻을 수 있도록 확장할 수 있다. 현재의 Context Toolkit 에서는 센서들이 얻게 되는 정보들을 특별한 여과 없이 그대로 어플리케이션이 얻도록 하고 있다. 하지만 widget 에게 좀 더 AI 를 부여하여 과거의 데이터를 토대로 미래의 일을 예측하는 일, 혹은 에러의 발생을 탐지하고 보정하는 작업등을 이룰 수가 있다.

이러한 신뢰성을 높이기 위한 방법들을 제공하기 위해 모델을 수립하고 검증할 수 있을 것이다. 추후의 과제에서 이러한 모델을 수립하게 될 것이다[7,8].

현재 DWB 모듈의 확장성에 대한 연구를 하고 있으며 보다 세밀한 작업이 가능하도록 기존의 프로토콜을 수정하는 방안을 연구하고 있다. 추후에는 좀 더 개선된 프로토콜을 제안할 것이며, 단순한 기능을 하는 DWB 가 아닌 보다 동적인 환경에 적합하도록 개선하는 것이 우리의 초점이 될 것이다.

참고문헌

[1] Anind K. Dey & Gregory D. Abowd, “The Context Toolkit : Aiding the Development of Context – Aware Applications”

[2] Anind K. Dey, “Enabling the Use of Context in Interactive Applications”

[3] Stephen S. Yau, Deepak Chandrasekar, Dazhi Huang, “An Adaptive, Lightweight and Energy-Efficient Context Discovery Protocol for Ubiquitous Computing Environments”

[4] Anind K. Dey & Gregory D. Abowd, “A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications”

[5] http://www.cc.gatech.edu/fce/contexttoolkit/documentation/tutorial/index.html, “Context Toolkit Manual”

[6] Anind K. Dey, Jennifer Mankoff, Gregory D. Abowd and Scott Carter, “Distributed Mediation of Ambiguous Context in Aware Environments”

[7] Anind K. Dey and Gregory D. Abowd, “Towards a Better Understanding of Context and Context-Awareness”

[8] “ http://www.upnp.org/download/draft_cai_ssdp_v1_03.txt”, SSDP(Simple Service Discovery Protocol)