맥락 인식 애플리케이션을 위한 사용자 중심의 맥락 모델*

홍동표, 우운택
광주과학기술원 U-VR 연구실
e-mail : {dhong, wwoo}@gist.ac.kr

User-centric Context Model for Context-aware Application

Dongpyo Hong, Woontack Woo
GIST U-VR Lab.

요 약

본 논문에서는 맥락 인식 (context-aware) 애플리케이션 개발에 있어서 사용자의 맥락 정보를 보다 효과적으로 처리할 수 있는 사용자 중심의 맥락 모델을 제안한다. 유비쿼터스 컴퓨팅 개념의 확산과 함께 맥락 인식 애플리케이션들에 관한 많은 연구가 진행되고 있다. 하지만, 맥락에 대한 정의는 여전히 모호하며, 애플리케이션들 마다 서로 다른 형태의 맥락을 활용하고 있기 때문에, 맥락에 대한 보다 구체적인 정의와 다양한 애플리케이션에 활용 가능한 형태의 맥락 모델이 필요하다. 제안된 사용자 중심의 맥락 모델에서는 사용자가 애플리케이션과 상호작용할 때 사용자의 직접적인 명령을 제외한 사용자와 관련된 정보를 맥락으로 정의한다. 또한, 제안된 사용자 중심의 맥락은 5W1H 형태로 구조화한 맥락요소 (ContextElement), 맥락 요소들을 편리하게 처리할 수 있는 연산자들을 포함하는 맥락 (Context), 그리고 단편적인 맥락 정보뿐만 아니라 기존의 맥락 정보까지도 활용할 수 있는 맥락메모리 (ContextMemory)로 구성된다. 특히, 다양한 센서들로부터 획득된 정보를 맥락 모델의 인터페이스를 통해서 맥락 인식 애플리케이션에서 활용할 수 있기 때문에, 서로 다른 맥락 인식 애플리케이션들을 개발함에 있어서도 동일한 맥락 모델을 사용할 수 있는 장점이 있다. 제안된 맥락 모델의 유용함을 보이기 위해서, 센서로부터 획득된 맥락 정보를 처리하는데 소요되는 시간을 측정하는 실험을 하였다. 따라서 제안된 사용자 중심의 맥락 모델은 사용자와 맥락 인식 애플리케이션간 자연스러운 상호작용을 지원할 것으로 기대된다.

1. 서론

유비쿼터스 컴퓨팅 개념의 확산으로 다양한 분야에서 맥락 (Context)을 활용하는 연구가 활발하게 진행되고 있다[1,2]. 그리고, 사용자 상호작용(HCI: Human Computer Interaction)분야에서는 보다 똑똑한 시스템을 구현하기 위해서 사용자의 직접적인 입력뿐만 아니라 간접적인 정보 (예, 사용자의 성별, 나이, 위치 등)도 활용할 수 있는 기술들이 연구되어왔다[3,4]. 또한, 맥락 인식 (Context-aware) 애플리케이션 개발에 있어서도 다양한 센서를 활용하는 연구에서부터 맥락 인식 애플리케이션을 위한 프레임워크 연구까지 다양한 연구가 진행되고 있다[5,6]. 하지만, 사용자 중심의 맥락 인식 애플리케이션을 개발하기 위해서는 사용자의 맥락 정보를 구조적으로 기술할 수 있는 맥락 모델링 기법이 요구된다.

기존의 맥락 모델과 관련된 연구에서는 특정 애플리케이션에서만 사용 가능한 맥락만을 기술할 수 있는 다소 제한된 형태였기 때문에 다양한 맥락 애플리케이션에 공통으로 활용하기에 어려움이 있었다[7]. 그리고, 맥락 모델링에 있어서도 특정 시점의 맥락만을 표현하였기 때문에, 애플리케이션 개발자들이 별도의 맥락 저장소와 같은 요소들을 직접 개발해야 하는 불편함도 있었다.

본 논문에서는 맥락 인식 애플리케이션 개발에 있어서 사용자의 맥락 정보를 보다 효과적으로 처리할 수 있는 사용자 중심의 맥락 모델을 제안한다. 우선, 제안된 맥락 모델을 위해서 사용자 중심의 맥락을 정의하고, 정의된 맥락을 효과적으로 표현하고 활용할 수 있도록 계층적 구조로 맥락 모델을 구성한다. 사용자 중심의 맥락 모델은 5W1H 형태로 구조화한 맥락요소 (ContextElement), 맥락 요소들을 편리하게 처리할 수 있는 연산자들을 포함하는 맥락 (Context), 그리고 단편적인 맥락 정보뿐만 아니라 과거의 맥락 정보도 활용할 수 있는 맥락메모리 (ContextMemory)로 구성된다. 맥락요소는 제안된 맥락 모델에서 맥락 정보를 표현하는 최소 단위이며, 애플리케이션 개발자들이 필요에 따라서 언제던지 필요한 맥락 정보를 추가, 삭제할 수 있는 기능을 제공한다. 그리고, 제안된 맥락

*본 연구는 21 세기 프론티어 연구 개발 사업의 일환으로 추진되고 있는 정보통신부의 유비쿼터스 컴퓨팅 및 네트워크 원천 기반 기술 개발사업의 지원에 의한 것임

모델에서 맥락은 맥락 애플리케이션 개발에 있어서 빈번하게 사용되는 맥락 정보들간의 비교나 통합 등을 쉽게 할 수 있도록 관련된 연산자들을 제공한다. 맥락메모리는 단편적인 맥락 정보뿐만 아니라 이전의 맥락 정보도 활용할 수 있도록 일련의 맥락 정보들을 관리할 수 있는 기능을 제공한다. 끝으로, 제안된 사용자 중심의 맥락 모델은 맥락 모델이 제공하는 인터페이스를 통해서 다양한 센서들로부터 획득된 정보를 활용할 수 있기 때문에, 서로 다른 맥락 인식 애플리케이션들을 개발함에 있어서도 동일한 맥락 모델을 사용해서 개발할 수 있는 장점이 있다.

본 논문의 구성은 다음과 같다. 2 장에서는 제안된 맥락 모델을 위한 사용자 중심의 맥락 정의와 구조 그리고 각 요소들의 특징들에 대해서 살펴본다. 3 장에서는 제안된 맥락 모델을 이용한 맥락 인식 애플리케이션을 통한 실험 결과를 보이고, 4 장에서는 향후 연구 과제에 대해 언급함으로써 결론을 맺는다.

2. 사용자 중심의 맥락 모델

2.1 사용자 중심의 맥락 정의와 맥락 모델 구조

제안된 사용자 중심의 맥락 모델에서는 맥락을 다음과 같이 정의한다[7].

"사용자 중심의 맥락이란 사용자가 관심 있는 응용 서비스 (콘텐츠)와의 상호작용에 영향을 미치는 사용자의 상태에 대한 묘사이다."

사용자 중심의 맥락을 위와 같이 정의함으로써, 사용자 중심의 맥락 모델은 사용자의 맥락 정보를 애플리케이션에서 보다 구조적으로 활용할 수 있도록 그림 1 과 같이 구성된다.

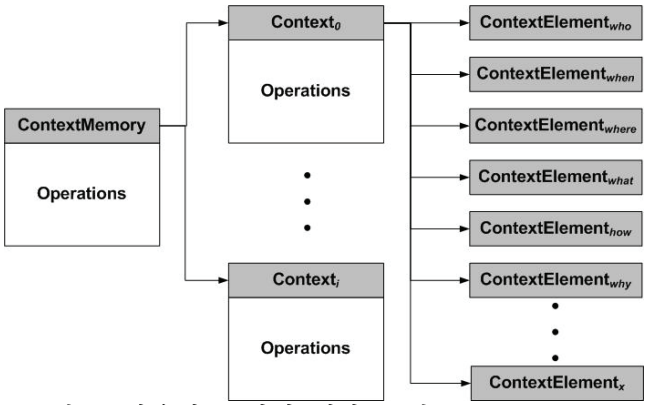

그림 1. 사용자 중심의 맥락 모델 구조

그림 1 에서 보듯이, 제안된 맥락 모델의 최소 단위는 **ContextElement** 이며 5W1H (WHO, WHEN, WHERE, WHAT, HOW, WHY)로 이루어진다 (표 1 참조). **Context** 는 **ContextElement** 들의 리스트로 이루어지며, **ContextElement** 간의 비교 · 검색 · 정렬과 같은 연산이 가능하다. **ContextMemory** 는 맥락을 시간 순서로 저장하고 관리하는 역할을 담당한다. 애플리케이션 개발자는 **ContextMemory** 의 인터페이스를 통해서 필요한 맥락을 추가하거나 추출한다.

표 1. 맥락 요소의 의미와 그 적용 예

맥락 요소	의미	예
WHO	사용자의 기본적인 정보	이름, 생일, 성별 등
WHEN	사용자가 특정 대상과 상호작용할 때의 시간	시/분/초, 년/월/일, 계절 등
WHERE	사용자의 위치	좌표, 장소, 지역 등
WHAT	사용자가 관심 있어하는 대상 (서비스)	*대상 (서비스)에 따라 종속적임*
HOW	사용자의 행동, 행위	멈춤, 걷기, 뛰기 등
WHY	사용자의 감정 상태	기쁨, 슬픔 등

2.2 ContextElement

ContextElement 는 제안된 사용자 중심의 맥락 모델에서 사용되는 최소 단위이며 기본 타입이다. 특히, 제안된 맥락 모델을 서로 다른 맥락 인식 애플리케이션에서도 활용할 수 있도록 디자인하기 위해서, 그림 2 와 같이 **ContextElement** 를 구조화하였다.

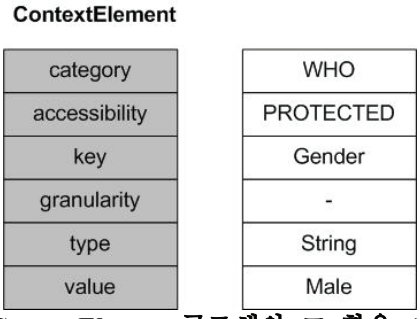

그림 2. ContextElement 구조체와 그 활용 예

그림 2 에서 보듯이, **ContextElement** 는 임의의 맥락 정보 기술이 가능하도록 6 개의 필드로 구성되며, 개발자가 필요한 맥락 정보를 임의로 할당하여 사용할 수 있다. 표 2 는 **ContextElement** 의 각 필드에 대한 의미이다.

표 2. 맥락 요소 구조체의 필드 및 의미

ContextElement	의미
category	5W1H 중 하나 지정 (WHO, WHEN, WHERE, HOW, WHY)
accessibility	맥락 정보의 공개 여부 (PUBLIC, PROTECTED, PRIVATE)
key	맥락 정보의 변수 이름
granularity	맥락 정보의 변수가 갖는 단위
type	맥락 정보의 변수가 갖는 타입
value	맥락 정보의 변수가 갖는 실제 값

ContextElement 는 또한 기본적인 연산자를 제공할 뿐만 아니라 개발자가 직접 추가할 수 있다. 예를 들면, 다음과 같이 "동등 (equality) 연산자"를 개발자가 원하는 형태로 정의할 수도 있다.

```
int operator== (const _ContextElement& ce1) {
  if ((m_Category==ce1.m_Category) &&
      (m_Accessibility==ce1.m_Accessibility) &&
        (m_Key==ce1.m_Key) &&
        (m_Type==ce1.m_Type) &&
        (m_Granularity==ce1.m_Granularity) &&
        (m_Value==ce1.m_Value))
     return true;
  else
     return false;
}
```

2.3 Context

Context 는 **ContextElement** 들을 보다 효과적으로 관리할 수 있는 데이터 구조이다. 특히, **Context** 는 **sort**, **merge**, **unique** 와 같은 연산자를 제공함으로써, 센서로부터 획득된 **ContextElement** 를 통합할 때 혹은, 특정 맥락 정보만 처리하고자 할 때 유용하다. 물론, **sort** 연산자를 활용하기 위해서는 **ContextElement** 에 order 연산자를 정의하거나 기본적으로 제공되는 것을 사용할 수 있다. 뿐만 아니라, 제안된 **Context** 는 특정 **ContextElement** 들을 최대한 빨리 찾기 위해서 **find** 와 같은 연산자도 제공한다. 그림 3 은 **Context** 의 내부 구조를 나타내는 그림이다.

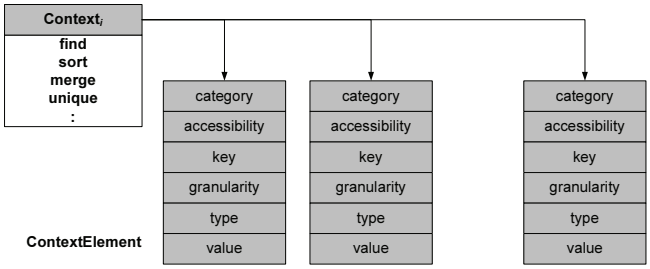

그림 3. 맥락의 연산자 및 내부 구조

그림 3 에서와 같이, **Context** 는 **ContextElement** 들을 리스트로 관리하기 때문에, 불필요한 **ContextElement** 를 저장할 필요가 없으며 새로운 **ContextElement** 가 추가 · 삭제되더라도 강건한 구조를 갖는다[8]. 즉, 제안된 **Context** 의 구조는 다양한 맥락 정보를 획득하거나 처리 관리하는데 유용하다.

3.4 ContextMemory

일반적으로 맥락 인식 애플리케이션들에서는 맥락 정보를 활용할 때, 한 시점의 맥락 정보만을 활용하는 경우가 많다. 하지만, 맥락 인식 애플리케이션이 사용자를 이해하고 사용자의 특성에 맞게 적절한 반응을 하기 위해서는 한 시점의 맥락 정보보다는 일정한 기간 동안 맥락 정보의 활용이 필요하다. 따라서, 제안된 **ContextMemory** 은 시간 순서에 따라 맥락을 저장하고, 특정 구간의 맥락을 임의로 접근할 수 있다. 그림 4 는 제안된 **ContextMemory** 의 구조이다.

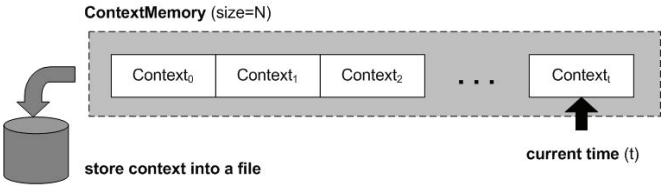

그림 4. ContextMemory 의 구조

그림 4 에서와 같이, 임의의 맥락은 시간 순서에 따라 하나씩 차례대로 추가되며, 특정 기간 동안 맥락을 유지한다. 그리고, 특정 기간 동안 모아진 맥락은 외부 파일 형태로 저장하도록 하여 필요한 맥락 정보만을 메모리 상에서 활용할 수 있다. 특히, 맥락 인식 애플리케이션의 특성에 따라 여러 모듈들이 존재할 수 있고, 각 모듈들이 동시에 **ContextMemory** 를 접근하려고 할 때, 자원 공유를 원활하게 하기 위한 뮤텍스 (Mutex) 를 사용한다. 또한, **ContextMemory** 가 갱신되는 시점을 이벤트로 처리하여 각 모듈은 언제 **ContextMemory** 에 접근해야 하는지 알 수 있다.

3. 실험 결과

제안된 맥락 모델의 유용성을 검증하기 위하여 그림 5 와 같이 실험을 구성하였다. 표 3 은 실험에 사용된 센서와 센서 입력을 처리하기 위해 사용한 소프트웨어들에 대한 설명이다[9,10].

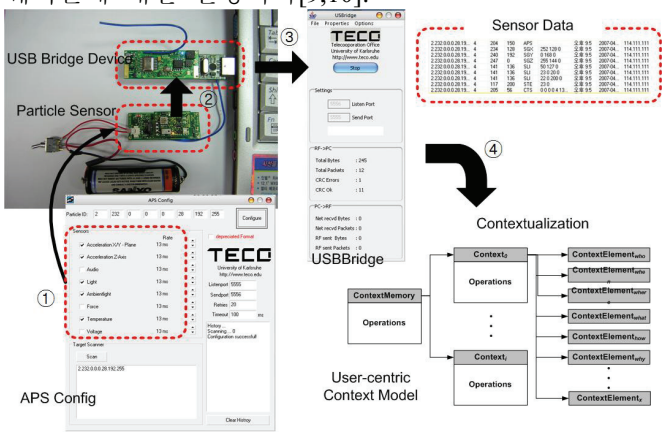

그림 5. 제안된 맥락 모델을 위한 실험 구성도

그림 5 에서 보듯이, Particle 센서로부터 획득된 데이터들은 USB Bridge 장치로 전달된다. 이때, 그림 5 의 ①과 같이 체크된 항목에 해당하는 데이터들만 Particle 센서로부터 USB Bridge 장치로 전달된다. USB Bridge 소프트웨어를 통해서 전달된 센서 데이터를 활용할 수 있다. 전달된 센서 데이터의 형태는 그림 5 의 ③과 같다. 이와 같이 센서로부터 획득된 데이터는 제안된 맥락 모델의 **ContextElement** 형태로 변환된다. 그리고 변환된 **ContextElement** 는 **Context** 의 요소로 추가되며, **ContextMemory** 에 순차적으로 저장된다.

표 3. 실험에 사용된 HW 와 SW 에 대한 기능 설명

항목		기능 및 사양
하드웨어	**Particle**	XYZ 축에 대한 움직임, 소리, 조도, 그리고 온도 측정 가능한 센서
	USB Bridge	Particle 센서로부터 획득된 데이터를 PC 에서 수신하거나 데이터를 particle 로 송신할 때 사용
	PC 사양 (OS, CPU, RAM)	실험에 사용한 PC는 Windows XP (Service Pack2), Intel Pentium Mobile 1.20GHz, 램 504MB
소프트웨어	**C++/STL**	제안된 맥락 모델을 구현하기 위해 사용된 프로그래밍 언어와 자료구조
	USB Bridge	Particle 센서로부터 PC 혹은 PC 로부터 Particle 센서로 데이터를 전달할 때 사용되는 소프트웨어. UDP 사용. (그림 5 ① 참조)
	APS Config	일반 사용자들이 손쉽게 Particle 센서를 사용할 수 있도록 도와주는 소프트웨어 (그림 5 ③ 참조)
	gprof	제안된 맥락 모델의 각 연산자들의 수행 속도를 체크하기 위한 소프트웨어

특히, 맥락 인식 애플리케이션 개발에 있어서 사용자의 맥락을 인식하고 실시간으로 반응하기 위해서는

맥락 정보의 실시간 획득, 통합, 관리가 중요하다. 따라서, 제안된 사용자 중심의 맥락 모델의 실시간성을 검증하기 위해서 제안된 맥락 모델에서 제공하는 연산자들 (**find, sort, merge, unique**)에 대해서 맥락 정보의 량에 따른 각각의 연산 시간을 측정하였다. 하지만, 센서로부터 획득된 데이터는 매번 다르기 때문에 제안된 맥락 모델의 연산자에 소요되는 시간을 보다 정확하게 측정하기 위해서 **ContextElement** 의 값(value)을 특정 범위에 위치하도록 보정하였다. 그림 6 은 GNU Profiler [11]를 이용하여 5 회 반복한 평균 값을 나타낸다.

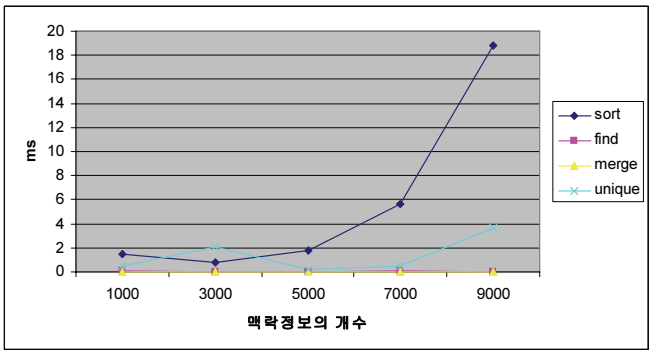

그림 6. 맥락 연산자들의 연산 시간

그림 6 에서 보듯이, 맥락 정보가 증가하더라도 제안된 맥락 모델에서 제공하는 연산자들은 평균 수 *ms* 의 연산 시간을 보장함을 알 수 있다. 특히, **sort** 와 **unique** 연산자의 경우에는 맥락 정보의 수가 증가할수록 연산 시간도 함께 증가하고 있다. 이는 센서로부터 획득된 데이터의 값이 점진적으로 증가하고 있음을 알 수 있다. 또한, 연산자 **find** 와 **merge** 의 경우에는 맥락 정보의 수와 관계없이 일정한 연산 시간을 보장하는데, 이는 센서로부터 들어오는 데이터들의 값이 각각 다름을 알 수 있다.

표 4 는 제안된 맥락 모델의 각 요소들이 사용하는 메모리 사용량을 바이트 (Byte)단위로 나타낸 것이다.

표 4. 메모리 사용량

Type	Compile-time Size	*N (개수)*	Run-time Size
ContextElement (CE)	24	*1*	*24*
Context (C)	16	*0 < #CE*	*16 * #CE*
ContextMemory (CM)	32	*0 < #C < N*	*32 * #C*

표 4 에서처럼 **ContextElement** 는 시간에 상관없이 동일한 크기의 메모리를 사용하지만, **Context** 의 경우에는 **ContextElement** 의 개수에 따라 메모리 사용량이 가변적임을 알 수 있다. 그리고, **ContextMemory** 는 일정 시간이 경과한 후에는 항상 동일한 메모리 량을 사용함을 알 수 있다. 이는 **ContextMemory** 는 일정 개수 (*N*)의 **Context** 만을 메모리에서 관리하고 이전 **Context** 들은 파일 형태로 저장하기 때문이다.

따라서, 제안된 사용자 중심의 맥락 모델은 실시간성을 보장하면서도, 맥락 정보에 상관없이 일정한 메모리만을 사용하기 때문에 맥락 인식 애플리케이션에 충분히 활용 가능한 모델이다.

4. 결론 및 추후 과제

본 논문에서는 맥락 인식 애플리케이션 개발을 위해 사용자의 맥락 정보를 효과적으로 관리할 수 있는 사용자 중심의 맥락 모델을 제안하였다. 제안된 사용자 중심의 맥락 모델은 **ContextElement**, **Context**, 그리고 **ContextMemory** 를 통해서 센서로부터 획득된 사용자의 맥락 정보를 저장 · 통합 · 처리를 효과적으로 할 수 있었다. 뿐만 아니라, 제안된 맥락 모델은 다양한 맥락 정보를 쉽게 추가할 수 있도록 확장 성을 고려하여 설계되었다. 추후 과제로는 제안된 맥락 모델을 보다 다양한 맥락 인식 애플리케이션에 적용하여, 맥락 인식 애플리케이션 개발에 충분한 확장 성을 제공하는지에 대한 검증이 필요하다. 그리고, 제안된 맥락 모델을 맥락 인식에 보다 쉽게 활용될 수 있도록 SQL 과 같은 쿼리기능에 대한 연구도 필요하다.

참고문헌

[1] G. Chen, D. Kotz. “A Survey of Context-Aware Mobile Computing Research,” Technical Report TR2000-381, November, 2000.

[2] K. Mitchell, “A Survey of Context-Awareness,” Internal Technical Report, Lancaster University, Computing Department, March 2002.

[3] B.N. Schilit, N.L. Adams, R.Want, “Context-aware computing applications,” In IEEE Workshop on Mobile Computing Systems and Applications, 1994.

[4] A. Schmidt and K.V. Laerhoven, “How to build smart appliances,” IEEE Personal Communications, Special Issue on Pervasive Computing, vol. 8, no. 4, pp. 66-71, August 2001.

[5] A. Dey, D. Salber, G. Abowd, “A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications,” Human-Computer Interaction (HCI) Journal, vol. 16, no. 2-4, pp. 97-166, 2001.

[6] 홍동표, 신춘성, 오세진, 우운택, “사용자-콘텐츠-환경간의 이음매없는 상호작용을 위한 프레임웍,” SK 텔레콤 Telecommunications Review, vol. 16, no. 4, pp. 644-661, 2006.

[7] 홍동표, 우운택, “맥락의 정의와 모델링에 관한 연구 동향 및 전망,” 한국멀티미디어학회지(KMMS), vol. 10, no. 1, pp. 149-162, 2006.

[8] N. M. Josuttis, “The C++ Standard Library - A Tutorial and Reference,” Addison-Wesley, 1999.

[9] http://particle.teco.edu/device/index.html

[10] http://particle.teco.edu/software/index.html

[11] http://www.gnu.org/software/binutils/manual/gprof-2.9.1/gprof.html