유비쿼터스 컴퓨팅 환경에서 사용자 생활패턴을 고려한 상황정보 생명주기 관리 메커니즘

장현준[1],박인석[2],현순주[3],이동만[4]
한국정보통신대학교[1 2 3 4]
{comma78[1],ispark[2],shyun[3],dlee[4]}@icu.ac.kr

Context Life Cycle Management Mechanism considering User Life Pattern in Ubiquitous Computing Environments

Hyunjun Chang[1], Insuk Park[2], Soon J. Hyun[3], and Dongman Lee[4]
Information and Communications University[1 2 3 4]

요 약

유비쿼터스 컴퓨팅 환경에서 상황정보 관리자는 특정인 주변의 상황정보의 변화를 관찰하여 그 사람의 현재 상태를 추론하는 역할을 한다. 지금까지의 상황정보 관리자는 사용자의 상태를 추론하는 과정에서, 관련된 상황정보 값들의 변화를 곧바로 사용자 상태의 변화로 간주하기 때문에, 사용자의 의도와는 상관없이 빈번하게 사용자 상태가 변화되는 결과가 초래되었다. 다시 말하여, 실제 사용자가 현재의 추론된 상태를 계속 유지하고 싶은 의도를 지닌 경우에도, 주변의 특정 상황정보의 값이 규칙에서 규정한 것과는 다르게 조금이라도 변하게 되면, 현재까지 유지되던 사용자의 상태 상황정보가 새로이 추론된 사용자상태 상황정보로 대체되게 되는 것이다. 따라서 해당 사용자는 전 상태를 기반으로 받고 있던 서비스를 더 이상 제공받지 못하게 되는 문제가 생기게 된다. 본 논문에서는 실제로 지속될 필요가 있는 것과 지속될 필요가 없는 상황정보를 구분하여 사용자의 상태 상황정보를 관리하고 범용적으로 적용될 수 있는 메커니즘을 제안한다. 본 메커니즘을 적용하게 되면, 사용자의 상태 상황정보의 존립에 영향을 미치는 주변 상황정보의 값이 달라지게 되어도, 활성화되어 있던 당시의 사용자의 상태정보는 '보류'된 상태로 남아있다가 활성화시키는 이벤트가 발생하였을 때에 '재개'되어 보류되기 전까지 서비스의 상태 그대로, 사용자의 생활패턴과 의도에 부합되는 서비스를 지속적으로 제공할 수 있게 된다.

Keyword : Context life cycle, User life pattern, Context awareness, Context management

1. 서 론

유비쿼터스 컴퓨팅 환경은 현재 사용자의 상황을 기반으로 의도를 파악하여, 사용자에게 적절한 서비스 내지는 정보를 제공하여 주는 자동화된 환경이다. 그러므로 사용자는 서비스를 받기 위해 일부러 주의를 전환하지 않고서도 현재 활동에 계속하여 집중하며 서비스를 받을 수 있게 된다. 사용자의 의도를 파악하기 위하여 여러 가지 상황정보들을 활용하게 되는데, 대표적인 것으로는 센서를 통하여 센싱된 상황정보들 (예, 사용자의 위치 또는 자세)과 미리 정의된 상황정보 (예, 사용자의 성별 또는 나이)를 들 수 있다. 이러한 두 가지 상이한 상황정보는 서로 각기 다른 시간적 특성을 갖는데, 보통 센싱된 상황정보는 비교적 짧은 지속특성(즉, 생성되고 사라지는 주기가 짧음)을 지닌 반면에, 정의된 상황정보는 비교적 오래 지속되는 특성을 지닌다. 기존의 상황정보 관리 시스템은 이러한 상황정보를 이용하여 적절한 서비스를 제공하기 위한 사용자의 상태정보를 파악함에 있어서, 사용자 상태를 나타내는 상황정보 그 자체에 대한 시간적 특성은 전혀 고려하지 않고 있다는데 문제가 존재한다. 문제를 좀 더 명확히 하기 위해, 기존의 이벤트라는 것을 세분화해 정의

한 연구 논문[12]을 살펴보면, 해당 논문에서는 이벤트(event)라는 것을 크게 시간적으로 짧은 순간에만 지속하는 특성을 지닌 '순간 이벤트(instant event)'와 비교적 일정 시간 동안 지속하는 특성을 지닌 '간격 이벤트(interval event)'로 나누고 있는 것을 살펴볼 수 있다. 이러한 정의를 유비쿼터스 환경의 상황정보 모델링 방식에 적용해서 생각해 볼 수 있다. 즉, 만약 어떤 사용자 상태정보가 회사에서 회의를 하고 있다거나, 집에서 잠을 자고 있다고 하는 '간격 상황정보'적 특성을 가진 것일 경우, 이러한 사용자 상태정보는 그렇지 않은, 즉 집에 들어왔다 또는 회사에서 퇴근했다라고 하는 일시적인 특성을 지니는 '순간 상황정보'에 비해서 비교적 일정 시간 동안 지속, 유지해줄 필요가 있다는 것이다. 이러한 서로 다른 시간적 특성을 지니는 상황정보들을 사용자의 상태를 추론하고 유지하는 과정에서 고려하지 않게 되면, 주변의 자주 변하는 특성을 지닌 (즉, 시간 주기가 짧은) 센싱된 상황정보의 값의 변화로 인해, 지속될 필요가 있는 사용자 상태정보가 다른 상태 정보로 대체되게 되어 그 때까지 받고 있던 서비스를 지속하여 받지 못하게 되는 문제가 생기게 된다. 즉, 지속적으로 제공해야 할 필요가 있는 '간격 상황정보'들을 위해 제공되던 해당 서비스가 완전히 철회되었다가, 일정 시간 후, 센싱된 정보가 원래 값을 회복함에 따라 동일한 서비스가 처음부터 다시 시작되는 번거로운 과정이 반복되게 되는 것이다.

이와 비슷한 문제를 다룬 관련한 연구 논문 [5]을 살펴보면, 특정 사용자가 회사에서 회의를 하고 있는 경우를 예로 들고 있는 것을 볼 수 있다. 회의에 참여하고 있던 특정 사용자가 회의를 참여하겠다는 의도를 가진 채로, 일시적으로 회의장소를 떠나게 될 경우를 예로 들고 있는데, 해당 논문에서는 이러한 문제를 해결하기 위해 '가정 기반 추론(assumption-based reasoning)'이라는 추론 방식을 도입, 기존의 사용자 상태정보 추론 방식과 융합하여 문제를 해결할 것을 제시하고 있다. 그러나 이러한 문제 해결 방식은 위에서 언급한 것과 다른 사용자 상태 정보를 대상으로 비슷한 문제가 발생하였을 경우에, 그것을 해결하기 위한 보편적인 문제 해결 방식이 아닐 뿐만 아니라, 새로 부가된 가정 기반 추론 방식을 적용하기 위해서 필요한 또 다른 부가적인 정의된 상황정보들을 필요로 한다는데 문제가 존재한다. 무엇보다 추론 방식이 복잡해짐에 따라서 해결하기 위한 시간을 더욱 많이 필요로 한다는데 또 다른 문제가 있다.

본 논문에서는 새로이 상황정보 생명주기 관리 방식을 도입하여 이와 같은 문제를 해결하고자 한다. 기본적으로 [2][12]에서 제기한 이벤트 분류 방식 (순간 이벤트와 간격 이벤트)에 근거하여 문제 해결 방식에 적용하였고, 유비쿼터스 컴퓨팅 시스템 메커니즘에 적용하기 위하여 생명 주기 단계를 포함 기반 모델링방식은 온톨로지 서술 언어인 OWL [17]을 이용하여 정의하였다. 각 주기는 준비단계(ready), 활성단계(running), 보류단계(suspended), 재개단계(resumed), 만기단계(expired), 종결단계(terminated)의 6 단계로 세분화 하였으며, 이러한 각 생명주기의 논리적인 단계별 전환 과정을 또한 아울러 제시하였다. 이러한 상황정보 생명주기 관리 메커니즘은 Active Surroundings[11] 즉, 상황인지 서비스를 제공하기 위한 유비쿼터스 컴퓨팅 미들웨어 구조를 기반으로 구현되었다.

전체적인 논문의 전개 방식은 다음과 같다. 2 절에서는 관련 연구 논문들을 서술하였고, 3 절에서는 상황정보 생명주기 설계와 관련된 자세한 접근 방법에 대해서 논한다. 4 절에서는 구현 부분과 전체적인 시스템의 구조를, 5 절에서는 마지막으로 결론과 앞으로 더 해나가야 할 부분에 대해서 언급한다.

2. 관련 연구

연구 논문 [8]에서는 상황 정보의 시간적인 특성에 대해서 언급하고 있다. 즉, 센싱된 상황정보는 동적인 특성을 가지고 있어서 짧게는 몇 초 동안, 길게는 몇 시간 동안 지속되는 특성을 가지고 있는 반면에, 정의된 상황 정보는 정적인 특성을 가지고 비교적 오랜 시간 동안 지속되는 특징을 가지고 있다고 말하고 있다. 그러나 여기에서는

위의 두 가지 상황정보를 통하여 추론되게 되는, 사용자의 상태정보의 시간적인 특성에 대해서는 아무런 언급도 하고 있지 않다. 사용자의 상태정보는 곧바로 상황인지 응용 프로그램의 작동 방향을 결정짓는 중요한 요소이다. 그러므로, 이러한 사용자 상태정보의 시간적인 특성을 고려하지 않게 되면, 사용자 상태정보의 빈번한 변동으로 인하여 서비스가 불안정하게 제공되게 되는 문제가 발생하게 된다.

본 논문에서 제기한 것과 비슷한 문제를 제기하여 해결 방식을 제안한 논문[5]을 또한 살펴보면, 그들이 제시하는 문제 상황은 다음과 같다. 사용자가 정해진 어떤 특정한 시간 간격 동안 머물러 있어야 하는 상황 (즉, 예를 들어 회사에서 정해진 회의 시간에 참석하는 경우)에 있다가 잠시 회의가 이루어지는 공간에서 이탈 할 경우이다. 이러한 문제를 해결하기 위해서, 해당 논문[5]에서는 가정기반 추론방식(assumption-based reasoning)을 적용할 것을 제안하고 있는데, 이러한 추론 방식을 적용하게 되면, 사용자가 잠시 자리를 비우게 되는(즉, 사용자의 위치 센서의 값이 변하게 되는) 경우에라도 사용자의 상태정보 (여기서는, 회의 중 이라는 상태정보)는 지속적으로 유지시켜 줄 수 있게 된다는 것을 해당 추론방식에 따라 논리적으로 풀어 보여주고 있다. 그러나 이러한 방식은 위와 같은 특정한 한 경우를 해결하는 데는 도움이 될 지 모르지만, 비슷한 다른 문제를 해결하려고 하는 경우에는, 제안된 가정기반 추론방식을 적용 하기 위해 부가적으로 정의된 상황정보를 필요로 한다는데 문제가 있다. 또한 기존의 추론 방식과 더불어 복합적으로 가정기반 추론 방식을 적용해야 함으로 인해서 추론 과정이 복잡해지는 단점이 존재하게 된다.

3. 문제 해결 방법

3-1. 기존의 상황정보 관리 시스템에서의 상황정보 생명주기

상황정보의 생명주기에 대해서 보다 뚜렷이 이해하기 위해서는, 기존의 유비쿼터스 컴퓨팅 상황정보 관리 시스템에서 상황정보를 어떤 식으로 관리하는 지를 알아 보는 것이 필요하다. 본 논문에서 상황정보 생명주기를 논할 때의 상황정보는 사용자의 실제 의도와 연관된 시스템을 통해 추론된 사용자의 상태정보를 가리킨다.

기존의 상황정보 관리 절차는 크게 세가지 단계로 살펴볼 수 있는데, 첫 번째는, 어떤 이벤트가 발생하여 상황정보의 값이 변하기를 기다리는 단계이고, 두 번째는, 변화된 상황정보를 토대로 사용자의 의도를 파악하는 단계 (즉, 사용자의 상태정보), 마지막으로 세 번째는, 파악된 사용자의 상태정보를 기반으로 그에 적합한 서비스를 제공하는 단계이다. 그림 1 을 보면 그러한 상황정보 관리 절차를 순서도를 통해서 나타내어 주고 있다.

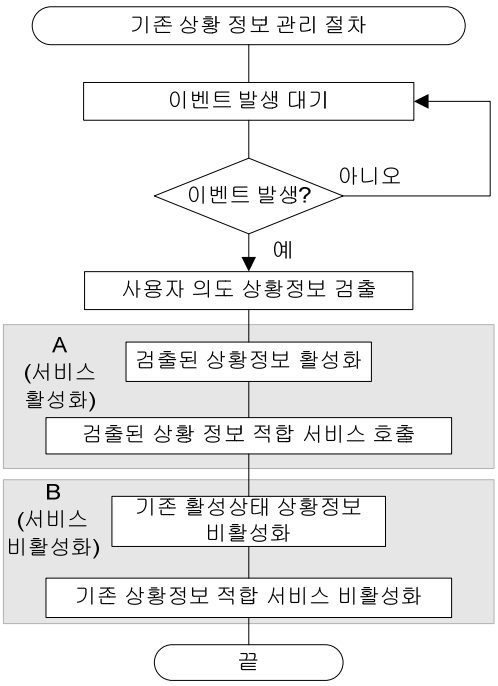

그림 1. 기존의 상황정보 관리 절차

그림 1 에서 사용자의 상태정보는 오직 활성화 상태와 비활성화 상태의 두 가지로 존재하고 있다. 활성화라는 것의 의미는 그것이 사용자의 현재 의도를 제대로 반영해주는 상태정보라는 것을 의미한다. 특정한 어떠한 이벤트의 발생으로 인하여, 그로 인한 사용자의 상태정보가 새로이 추론되게 되고, 기존의 활성화 되어 있던 사용자의 상태정

보는 비활성화되는 과정을 거치게 된다. 따라서 그에 따라 기존의 활성화된 사용자 상태정보에 맞게 제공되던 서비스의 제공이 철회되게 된다. 우리는 여기서, 사용자의 상태정보의 생명주기는 처음에 어떠한 식으로든 발생하는 이벤트에 밀접하게 연관되어 있음을 알 수 있다.

3-2. 상황정보 생명주기 관리 메커니즘 제안

기존의 상황정보 관리 방식은 사용자의 상태정보를 일정하게 유지시켜주지 못하는 단점을 가지고 있다는 것을 서론을 통해 언급한 바 있다. 예를 들어, 어떠한 사용자가 집에서 '목욕'을 하고 있으면서, 관련한 서비스를 제공받고 있다가, 잠시 동안 사용자의 위치가 변하거나, 온수기의 상태를 변화시켰을 경우의 이벤트가 발생함으로 인하여, 비록 사용자가 계속해서 그러한 상태를 유지하고자 하는 경우에도 불구하고, 사용자의 지속적인 '목욕'이라는 상태정보는 지속되지 못하게 되는 것이다.

우리는 연구 논문 [12]에서 제안하는 이벤트를 분류하는 방식, 즉 '순간 이벤트(instant event)'와 '간격 이벤트(interval event)'을 기반으로 문제를 해결하는 실마리로 삼고자 한다. '순간 이벤트'라고 하는 것은, 직관적으로, 이벤트가 발생한 앞뒤 사이에 시간적인 간격이 없는 것을 의미한다. 즉 예를 들어, 자동차의 사고 발생 또는 소포의 도착과 같은 것을 의미한다. '간격 이벤트'라는 것은 역시 직관적으로 보았을 때, 특정한 시간간격을 갖는 이벤트를 가리키는데, 예를 들어 2 시부터 3 시까지의 회의 시간 같은 것을 가리킨다. 유비쿼터스 환경에 적용해서 생각해 보면, '집에 들어왔다'거나 '회사에서 퇴근한다' 라고 하는 상황이 '순간 이벤트'에 속하는 것이고, '샤워 중'이라거나 '잠을 자고 있음'이라는 상황정보가 '간격 이벤트'에 속한다고 할 수 있다. 그러므로, '간격 이벤트'에 속하는 사용자의 상태정보가 유효한 상태일 경우 상황정보 관리 시스템은 그러한 상태정보를 일정 시간 간격 동안 지속적으로 유지해줄 필요가 있지만, 그렇지 않은 '순간 이벤트'에 속하는 것일 경우에는 지속해야 하는 시간에 대해서 고려할 필요가 없게 된다. 우리는 먼저 상황정보가 존재할 수 있게 되는 생명주기 단계를 크게 여섯 단계로 나누어보았다. 즉, 각 주기는 준비단계(ready), 활성단계(running), 보류단계(suspended), 재개단계(resumed), 만기단계(expired), 종결단계(terminated)의 6 단계이며 각각을 설명하자면 다음과 같다. 여기서 의미하는 상황 정보는 사용자의 상태를 나타내는 즉, 사용자의 의도를 반영한 사용자의 상태정보를 말한다. 준비단계(ready)는 아직 이벤트가 발생하지 않아 유효성을 검증 받기를 대기하는 단계이고, 활성단계(running)는 발생한 이벤트에 대해 유효하다고 검증된 단계를 의미한다. 보류단계(suspended)는 유효한 것으로 검증되어 활성화 되었다가, 또 다른 이벤트가 발생함으로 인하여 비활성화 되지 않고 계속 관련 서비스를 제공하여 주기를 대기하는 단계라고 할 수 있다. 재개단계(resumed)는 보류단계에 있던 상태정보가 다시 활성화 된 것을 말하며, 만기단계(expired)는 보류단계에 있다가 유효 시간이 만기 됨으로 인해서, 비활성화된 상태정보를 이르며, 종결단계(terminated)는 더 이상 유효하지 않은 상태정보로서 관련한 서비스 또한 철회되게 하는 상태정보를 말한다.

그림 2 를 살펴보면, 여기서 제안하는 상황정보 생명주기 메커니즘을 자세하게 알아 볼 수 있다. 이것은 크게 세 가지의 단계로 나누어 볼 수 있는데, 첫 번째 단계는 상황정보를 변화시키는 이벤트를 대기하고 있는 단계이고, 두 번째 단계는 이벤트가 발생함에 따라서 현재 상황정보의 생명단계를 고려하여, 보류 상태에 있던 것을 재개상태로 바꾸거나, 어떤 상태 종결 이벤트가 발생 (예를 들어, 'TV 를 본다'는 상태정보가 활성화된 상태에서 TV 가 꺼지게 되는 경우)하게 되는 경우 종결 상태로 전환시켜주는 단계를 나타낸다. 마지막 단계로는 유효하다고 검증된 사용자의 상태정보에 대해서 해당 서비스를 제공해주는 것과, '순간 이벤트'로 판명됨으로 인해서 기존에 제공되던 서비스를 종결시키는 단계이다.

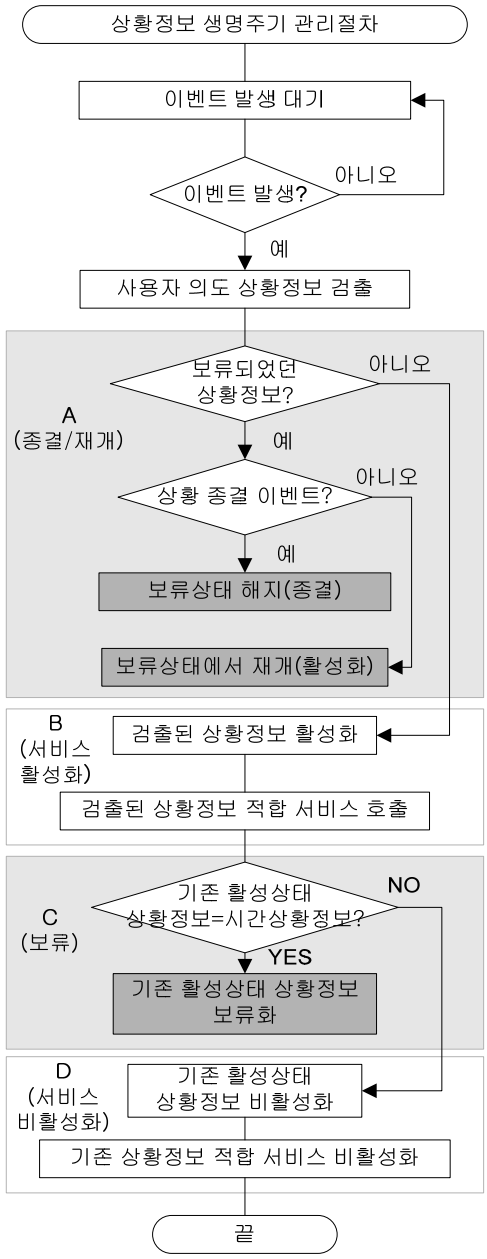

그림 2. 상황정보 생명주기 관리 방법제안

4. 구 현

본 논문에서 제안하는 상황정보 생명주기 메커니즘은, 전체적인 상황정보들을 관리하는 데 관여하는 유비쿼터스 컴퓨팅 시스템 미들웨어를 기반으로 구축되었다. 유비쿼터스 컴퓨팅 시스템 미들웨어는 센서를 통하여 발생한 이벤트를 받아들여, 그것들을 상황정보들로 변환시키고, 변환된 상황 정보들을 수집하여 사용자의 상태정보를 추론하는 모든 과정을 전담한다. 그러한 추론된 사용자의 상태정보는 그에 맞는 서비스를 제공하는 응용프로그램에게 또한 미들웨어를 통하여 알려지게 된다. 상황정보 미들웨어는 컨텍스트툴킷[7]과 제나[16]를 기반으로 JDK 1.5 버전을 통하여 구현되었다.

그림 3 의 왼쪽 부분은 상황정보를 관리하는 미들웨어를 구성하는 컴포넌트를 나타내고 있다. 크게 네 가지 컴포넌트로 구성되어 있는데, 상황정보 위젯은 센서로부터 받아들여진 이벤트를 상황정보로 변환하는 기능을 하게 되며, 상황정보 수집기는 각 위젯으로부터 상황정보를 수집하여, 사용자의 상태정보를 추론하는 역할을 맡는다. 또한 상황정보 해석기는 수집된 상황정보들을 해당 응용프로그램에게 알려주는 역할을 하게 된다.

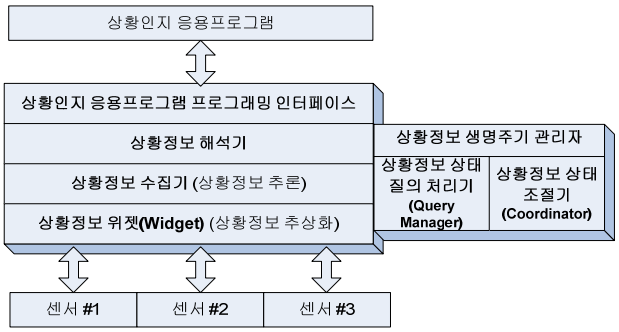

그림 3. 상황정보 관리와 상황정보 생명주기를 관리하는 전체적 시스템 구조도

그림 3 의 오른쪽 부분은 상황정보 생명주기를 관리하는 상황정보 생명주기 관리자를 나타내고 있다. 이것은 크게 상황정보 상태 질의 처리기와 상황정보 상태 조절기 두 가지의 컴포넌트로 이루어져 있는 데, 상황정보 상태 질의 처리기는 각 상황정보의 생명단계를 상황정보를 관리하는 상황정보 수집기나 상황정보 해석기에게 제공하는 역할을 하며, 상황정보 상태 조절기는 3-2 에서 제안한 방식대로 각 여섯 가지 상태 가운데 하나에 존재하는 상태정보들의 상태를 변화시키고, 조정하는 역할을 하게 된다.

5. 결론 및 앞으로의 과제

본 논문에서는, 사용자에게 보다 안정화된 서비스를 제공하기 위해 상황정보 생명주기를 관리하는 메커니즘을 제안하였다. 이벤트의 시간적인 특성을 고려한 분류방식에 기반하였으며, 이러한 분류 방식에 기반하여, 우리는 빈번한 상황정보의 변화로 인하여 사용자의 상태 정보가 변화되고, 그에 따라 사용자에게 제공되던 서비스가 완전히 철회되었다가 다시 처음부터 시작되는 문제점을 해결 할 수 있게 된다. 앞으로의 과제로는 실제 지능화된 가정에서 생활하는 사용자의 생활 패턴 정보를 수집하여 사용자의 생활 패턴에 일치된 상황정보 생명주기를 본 메커니즘에 적용하는 방안을 고려하고 자 한다.

6. 참고 문헌

[1] J. F. Allen, “Time and time again: The many ways to represent time,” Int'l. Journal of Intelligent Systems Vol. 6, Issue 4, Jul. 1991, pp. 341-356.

[2] J. F. Allen. “Actions and Events in Interval Temporal Logic,” Journal of Logic and Computation Vol. 4, Issue 5, Jul. 1994, pp. 531-579.

[3] H. Chen, T. Finin, and A. Joshi, “Semantic Web in the Context Broker Architecture,” Proceedings of the Second Annual IEEE International Conference on Pervasive Computer and Communications, Mar. 2004.

[4] H. Chen, T. Finn, and A. Joshi, “An Ontology for Context-Aware Pervasive Computing Environments,” Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review, May 2004.

[5] H. Chen, F. Perich, D. Chakraborty, T. Finin, and A. Joshi, “Intelligent Agents Meet Semantic Web in a Smart Meeting Room,” Proceedings of the Third International Joint Conference on Autonomous Agents & Multi Agent Systems (AAMAS 2004), Jul. 2004.

[6] A. K. Dey and G. D. Abowd, “Towards a Better Understanding of Context and Context-Awareness,” In Proceedings of the 2000 Conference on Human Factors in Computing Systems, The Hague, The Netherlands, Apr. 2000.

[7] A. K. Dey, G. D. Abowd, and D. Salber, “A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications,” HCI Journal, Vol. 16, Issue 2-4, 2001, pp. 97-166.

[8] T. Gu, H. K. Pung, and D. Q. Zhang, “A service-oriented middleware for building context-aware services,” Journal of Network and Computer Applications, Vol. 28, Issue 1, Jan. 2005, pp. 1–18.

[9] D. Garlan, D. Siewiorek, A. Smailagic, and P. Steenkiste, “Project Aura: Toward Distraction-Free Pervasive Computing,” IEEE Pervasive Computing, special issue on Integrated Pervasive Computing Environments, Vol. 21, Issue 2, Apr.-Jun. 2002, pp. 22-31.

[10] K. Henricksen, J. Indulska, and A. Rakotonirainy, “Modeling Context Information in Pervasive Computing Systems,” LNCS Vol. 2414, 2002, pp. 167–180.

[11] D. Lee, “Active Surroundings: A Group-Aware Middleware for Embedded Application Systems,” 28th Annual International Computer Software and Applications Conference (COMPSAC'04), 2004, pp. 404-405.

[12] F. Pan and J. R. Hobbs, “Time in OWL-S,” Proceedings of AAAI-04 Spring Symposium on Semantic Web, 2004.

[13] A. Ranganathan and R. H. Campbell, “An Infrastructure for Context-Awareness based on First Order Logic,” Journal of Personal and Ubiquitous Computing, Vol7, Issue 6, Dec. 2003, pp. 353-364.

[14] X. Wang, J. S. Dong, C. Y. Chin, S. R. Hettiarachchi, and D. Zhang, “Semantic Space: An Infrastructure for Smart Spaces,” IEEE Pervasive Computing, Vol.3, Issue 3, July-Sept. 2004, pp. 32-39.

[15] X. H. Wang, D. Q. Zhang, T. Gu , and H. K. Pung, “Ontology Based Context Modeling and Reasoning using OWL,” Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, Mar. 2004, p.18.

[16] Jena Semantic Web Framework,
“http://jena.sourceforge.net”

[17] Web Ontology Language (OWL),
“http://www.w3.org/2004/OWL/,”2004.