상황인지 서비스를 위한 컨텍스트 비교기 설계

김범수*, 박주열*, 조용윤* 최재영*
*숭실대학교 컴퓨터학과
e-mail:bskim@ss.ssu.ac.kr

Context Comparator Design for Context-Aware Service

Bum-Soo Kim*, Ju-Yeol Park*, Yong-Yoon Cho*, Jae-Young Choi*
*Dept of System Software, Soong-Sil University

요 약

유비쿼터스 환경에서는 동적으로 변하는 사용자의 상태정보를 바탕으로 상황에 맞는 서비스가 실시간으로 제공되는 상황인지 서비스를 지향한다. 이러한 상황인지 서비스는 실제 유비쿼터스 컴퓨팅 환경에서 주기적으로 센싱되어 들어오는 컨텍스트 정보에 대한 빠르고 정확한 컨텍스트 비교를 바탕으로 한다.

본 논문에서는 유비쿼터스 환경에서 발생하는 다양한 상황정보를 {주어, 동사, 목적어}형태의 RDF기반 컨텍스트 모델로 표현하고, 컨텍스트 유형별로 구분되는 컨텍스트 타입과 값의 튜플 조합을 컨텍스트 비교에 이용하는 상황인지 서비스를 위한 컨텍스트 비교기를 제안한다. 제안하는 컨텍스트 비교기는 센싱되어 들어오는 컨텍스트 정보를 컨텍스트 타입에 따라 세부 인덱스를 부여하여 컨텍스트 비교 횟수를 최소화할 수 있으며, 센싱된 입력 컨텍스트의 재활용 수준을 높일 수 있다. 따라서, 제안하는 컨텍스트 비교기는 동적인 유비쿼터스 환경에서 상황인지 서비스를 제공하기 기반 기술로써 위한 관련 응용프로그램 및 시스템 설계에 그 활용도가 클 것으로 기대된다.

1. 서론

유비쿼터스 컴퓨팅 관련 기술은 최근 다양한 형태로 생활환경 속에서 존재하며, 인간의 생활의 질적 개선과 편리함을 향상 시키고 있다. 유비쿼터스 컴퓨팅 환경(Ubiquitous Computing Environments)은 사용자 및 환경을 인식(Context aware)하고, 언제 어디서나 어떤 디바이스를 통해서도 끊이지 않는(seamless)서비스를 자동으로 제공하는 환경이다.[1]

유비쿼터스 컴퓨팅 환경에서의 상황인지 서비스는 인간의 간섭을 최소화하는 자동화 서비스를 제공하기 위한 상황인지 서비스 자동화 기술과 특정 공간에서 센싱되어 입력되는 컨텍스트(Context)정보를 사용자의 상황에 따라 올바르게 인지할 수 있는 컨텍스트 비교 관련 기술이 매우 중요하다.

비즈니스 프로세스 환경에서 성공적인 자동화 모델로 사용되었던 워크플로우 기술을 이용하여 상황인지 서비스를 제공하려는 연구가 현재 활발히 진행 중이다.[2]. 상황인지 서비스 제공을 위한 워크플로우 시스템은 사용자가 작성한 시나리오를 바탕으로 실제 유비쿼터스 환경에서 발생하는 실제 컨텍스트와의 비교에 따라 상황인지 서비스를 제공한다.[3]

따라서, 서비스를 제공하기 위해 센싱되어 입력되는 컨텍스트 정보가 서비스 시나리오에 기술된 조건에 따라 실행 가능한 컨텍스트 정보인지를 빠르고 정확하게 비교하는 컨텍스트 비교 알고리즘 기술이 중요하다. 보다 정확하고 빠른 상황인지 서비스를 제공하기 위해서는 컨텍스트 비교 과정에서 입력된 실제 컨텍스트의 재활용 수준을 높이고 컨텍스트 비교횟수를 최소화 할 수 있는 방법이 재고되어야 한다.

본 논문에서는 입력 컨텍스트의 재활용 수준을 높이고 컨텍스트 비교횟수를 줄이기 위한 컨텍스트 알고리즘을 포함하는 새로운 컨텍스트 비교기를 설계한다. 2장에서 관련연구에 대해, 3장에서는 비교 알고리즘을 제시하고, 4장에서는 기존사용 되는 컨텍스트 비교 알고리즘과 비교하여 그 성능을 테스트 한다.

2. 관련 연구

2.1 워크플로우 언어(Workflow language).

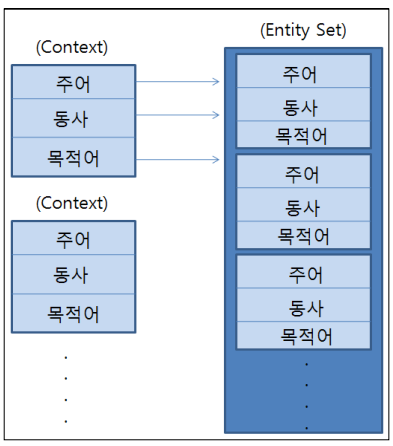
WfMC에서는 워크플로우를 "전체적인 또는 부분적인 비즈니스 프로세스의 자동화를 의미하며, 이때 문서, 정보, 태스크가 한 사용자에서 다른 사용자로 일련의 업무절차 규칙에 의한 처리를 위해 전달된다 [5]"라고 정의하고 있

다. 워크플로우는 수행되어야 하는 작업들 간의 수행 순서를 정의하며, 워크플로우가 실행 되면 작업들 간의 수행 흐름을 나타낸다. BPEL4WS과 같은 XML기반의 워크플로우 언어들은 유비쿼터스 환경에서 워크플로우의 서비스 간 분기조건을 충분히 얻을 수 없기 때문에 유비쿼터스 컴퓨팅 환경에서 발생하는 사용자의 컨텍스트 정보를 서비스 전이 조건으로 표현하고, 컨텍스트 비교를 통해 상황인지 서비스 제공을 지원하기 위한 워크플로우 언어로써 uWDL(ubiquitous Workflow Description Language)[4]이 제안되었다.

2.2 컨텍스트(Context).

컨텍스트는 장소, 사람이나 사물들을 구별 짓는 특징인 아이덴티티(identity), 사람이나 사물들을 포함하는 환경의 변화등으로 설명한다[6]. 유비쿼터스 컴퓨팅 환경에서 발생할 수 있는 사용자 상황정보 나 시나리오에서 표현 가능한 컨텍스트는 RDF 기반의 {주어, 동사, 목적어} 형태로 엔티티(entity)들로 표현 될 수 있다. 즉, 센서 네트워크로부터 제공되는 컨텍스트 정보는 RDF 기반의 컨텍스트 모델로 구성된 엔티티의 집합으로 객체화 된다[7].

유비쿼터스 환경에서 동적으로 센싱되어 들어오는 컨텍스트들을 엔티티 집합(Entity Set) 이라고 하고, 이러한 엔티티 집합에서 사용자에게 서비스를 제공하기위해 미리 시나리오에 정의 되어있는 컨텍스트와 비교과정을 거치게 된다. 그림 1은 uWDL 프레임워크에서 제공되는 컨텍스트 비교 알고리즘을 나타낸다. uWDL 컨텍스트 비교 알고리즘은 시나리오에 정의되어있는 컨텍스트들 및 센싱되어 들어오는 엔티티 집합과 순차적으로 하나씩 비교한다.



(그림 1) RDF 기반 컨텍스트 비교

그림 1의 비교과정에서 {주어, 동사, 목적어}가 일치하는 컨텍스트 정보가 들어왔을 때 서비스를 실행하게 된다. 그림 1의 uWDL 컨텍스트 비교 방법은 알고리즘의 단순성과 정확성에 있어서 장점을 가진다. 그러나 최상의 경우(best-case)에는 좋은 비교 횟수를 가지지만, 최악의 경우(worst-case)에는 비교 효율이 좋지 않다. 위 그림 1의 비교과정에서 n개의 컨텍스트, f개의 엔티티가 있다고 가정하면, 평균 비교회수는 (3n*f)/2 이 되기 때문에 시나리오에 정의 되어진 컨텍스트와 실제 센싱되어 입력되는 컨텍스트 정보가 많을수록 평균 비교횟수의 증가로 비교효율이 떨어진다.

유비쿼터스 컴퓨팅 환경에서 발생하는 실제 컨텍스트의 양은 매우 크며, 종류는 매우 다양하다. 따라서, 시나리오를 기반으로 하는 상황인지 워크플로우 시스템에서의 컨텍스트 비교 효율을 더욱 향상시키고, 동적인 상황인지 서비스를 제공하기 위한 새로운 컨텍스트 비교기가 요구된다. 새로운 컨텍스트 비교기는 센서로부터 사용자 정보를 실시간으로 전달 받아 비교 수행 시간 및 비교되는 횟수에 대해서 기존 컨텍스트 비교방법에 대해 효율성을 제공해야 한다.

3. 새로운 컨텍스트 비교기의 설계

유비쿼터스 환경에서 컨텍스트(Context) 비교는 사용자에게 센싱된 정보에 맞추어 원하는 서비스를 제공해 주기 위함이고, 이러한 컨텍스트의 비교는 빠르고 정확할수록 동적으로 변화는 유비쿼터스 환경에서 사용자에게 신뢰성 높은 서비스를 제공할 수 있다.

3.1 컨텍스트 모델(Context Model)

제안하는 컨텍스트 비교기가 사용하는 컨텍스트는 서비스 실행조건으로써 기술되는 상황정보와 실제 센서로부터 입력되는 상황정보에 대해 일관성 있는 구조적 컨텍스트 모델에 따라 객체화 된다. 다음 그림 2는 제안하는 컨텍스트 비교기가 이용하는 RDF기반의 컨텍스트 모델이다.

Context		
Subject	Subject Type	Person Type
	Subject Value	A
Verb	Verb Value	Situation
Object	Object Type	Conference Type
	Subject Value	presentation

(그림 2) 컨텍스트 모델

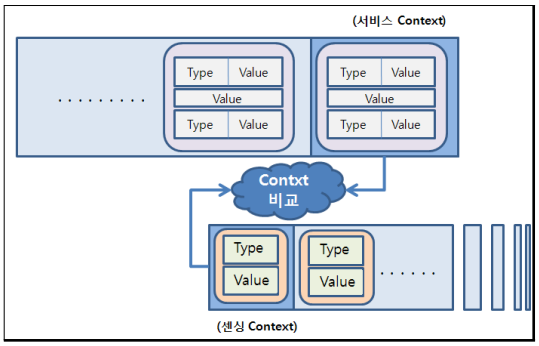
그림 2의 RDF 기반 컨텍스트 모델은 {주어, 동사, 목적어}의 트리플릿(triplet)으로 구성되며, 주어와 목적어는 컨텍스트 형(type)과 컨텍스트 값(value)으로 이루어진 순서쌍 데이터로 이루어진다. 동사는 값만으로 이루어진다. RDF 기반의 컨텍스트 모델은 서비스 도메인을 기술하기

위한 온톨로지와의 연동에 유연하며, 기존의 온톨로지 스토리지 및 미들웨어와의 연동에 유리하다. 또한, API 수준의 클래스 모델을 통해 쉽게 구현될 수 있다.

3.2 컨텍스트 비교 알고리즘.

컨텍스트(Context) 즉, 상황정보는 유비쿼터스 환경에서 실시간으로 센싱 되어 서비스를 하기 위해 적당한 컨텍스트 정보인지 확인하기 위해 비교 큐에 쌓이게 된다. 사용자는 동적인 움직임을 갖기 때문에 센싱되는 정보역시 계속해서 새로운 정보가 들어오게 된다. 이러한 환경에서 사용자에게 신뢰성 있는 서비스를 제공하기 위해서는 센싱되어 들어오는 정보의 손실이 없어야 하고, 센싱된 정보를 빠르게 비교 하여 어떠한 서비스를 제공할지 결정하는 과정이 중요하다.

기존의 시나리오 기반 상황인지 워크플로우 중 하나인 uWDL의 컨텍스트 비교 알고리즘의 경우에는 아래의 그림과 같은 비교 구조를 갖는다.



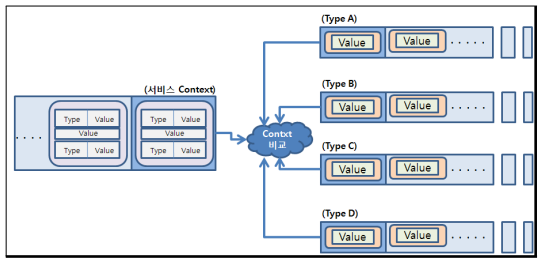
(그림 3) 컨텍스트 비교 과정.

위의 그림 3을 보면 하나의 큐에 센싱되는 컨텍스트 정보들이 저장되며, 이러한 센싱된 정보는 워크플로우 시나리오에 서비스 실행조건으로 정의되어 있는 컨텍스트 정보와 비교된다.

그림 3의 비교 알고리즘은 큐에 저장되는 컨텍스트의 주어(Subject)에 해당하는 컨텍스트 엔터티의 타입을 찾고, 일치하는 타입이 발견되면 그 값을 비교하여 시나리오에 기술된 주어 엔터티와 일치하는 센싱 정보인지를 결정한다. 동사와 목적어에 해당하는 컨텍스트 엔터티에 대해서도 같은 방법으로 컨텍스트 비교 과정을 수행된다. 이러한 방법에는 두 가지 단점이 존재하는데 하나의 대기 큐에서 순서대로 들어오는 정보를 순차적으로 비교하기 때문에 비교 횟수가 대기 큐와 시나리오에 기술된 컨텍스트 엔터티의 개수에 대해 상대적으로 많게 되는 점이고, 두 번째는 입력된 컨텍스트 엔터티가 시나리오에 기술된 컨텍스트 엔터티의 타입과 값, 모두와 일치하는지를 찾기 위해서 하나의 비교 대상에 대해 두 번의 비교 횟수를 갖는다는 점이다.

이러한 기존 컨텍스트 비교 알고리즘의 단점을 해결하기 위해서, 본 논문에서 제공하는 컨텍스트 알고리즘은 입력되는 컨텍스트에 대해 컨텍스트의 타입에 따라 개별적인 컨텍스트 대기 큐를 할당하는 방법을 사용한다. 즉, 새로운 컨텍스트가 센서로부터 입력되면 미리 만들어진 컨텍스트 대기 큐에 해당 컨텍스트 타입 큐가 있는지의 여부에 따라 새로운 컨텍스트 대기 큐를 생성하거나, 기존 같은 컨텍스트 대기 큐에 링크드 리스트 형태로 저장한다. 이를 통해, 동일한 컨텍스트 타입을 갖는 큐 안에서의 비교 횟수는 값만을 비교하게 되므로, 한 번의 비교횟수 만을 갖게 된다.

제안하는 알고리즘의 수행 과정은 아래의 그림과 같다.



(그림 4) 수정된 컨텍스트 비교 과정.

위의 그림 4에서 보이는 비교과정은 하나의 큐에서 비교했던 방법과 달리, 각 엔터티의 컨텍스트 타입을 비교하여 동일한 컨텍스트 타입을 찾으면 값만을 비교하는 일련의 과정을 갖는다. 이때, 새로운 대기 큐를 생성하는데 걸리는 시간은 컨텍스트를 비교하는 데에 걸리는 시간보다 매우 적으므로 그 효율성에 있어서 제안하는 컨텍스트 비교 알고리즘은 기존 방법에 비해 장점을 갖는다. 또한, 상황인지 워크플로우 서비스가 도메인은 물리적, 논리적으로 서로 밀접한 관련성을 갖는 한정된 종류의 컨텍스트 타입을 사용한다. 따라서, 제안하는 컨텍스트 비교 알고리즘이 적용되는 상황인지 서비스 응용프로그램은 제한된 서비스 도메인에서의 효율성을 가질 수 있다. 제안하는 컨텍스트 비교 알고리즘은 아래의 표1 및 표 2와 같다.

<표 1> 컨텍스트 비교 알고리즘1

입력 : 큐에 저장된 센싱된 정보 i
출력 : 결과를 나타내는 boolen 값

```
while(i.hasNext()) {
    CObject o = i.next();
    if(object.getType().equals(o.getType()))
    {
            if(object.getValue() == null) {
                return true;
            }
            else if(object.getValue()
                 .equals(o.getValue()) {
                return true;
            }
    }
}
return false;
```

위의 <표1>의 알고리즘은 큐에 저장된 데이터의 타입을 검사하고, 타입이 일치하는 경우에 값을 비교하는 과정을 거치게 된다. 아래의<표2>의 컨텍스트 비교 알고리즘2 는 표 1의 컨텍스트 비교 알로리즘1 을 통해 컨텍스트 엔터티의 타입과 값을 비교하는 과정에 있어서 컨텍스트 엔터티의 타입에 대한 대기 큐의 존재 여부에 따라 새로운 대기 큐의 생성 및 값을 비교한다.

<표 2> 컨텍스트 비교 알고리즘2

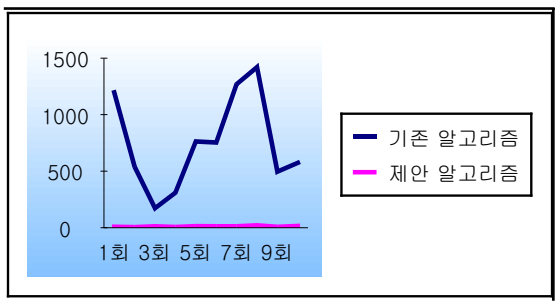
```
else if(hashMap.containsKey(object.getType()))
{
    while(i.hasNext()) {
        CObject o = i.next();
        if(object.getValue().equals(o.getValue()))
        {
            return true;
        }
    }
}
return false;
```

알고리즘1과 2는 컨텍스트 엔터티의 타입을 먼저 비교하여 타입이 일치하는 큐에 해당하는 값들만을 비교하기 위해 while loop를 수행한다. 즉, 큐에 저장된 하나의 센싱 정보를 비교하기 위해서는 일치하는 Type의 큐를 찾는 과정과 한 번의 컨텍스트 엔터티 비교과정인 값 비교를 거치게 된다.

4. 실험 및 평가

제안하는 컨텍스트 비교기를 통한 컨텍스트 비교 효율성을 실험하기 위하여 자바를 사용하여 임의의 문자열 값을 같는 10가지의 데이터 타입을 정의 해놓고 주어, 동사, 목적어에 맞게 10가지의 엔티티와 이와 같은 방식으로 구성된 10000개의 센싱 컨텍스트를 가상으로 설정한 후 기존 알고리즘 방식과 제안 알고리즘 비교 방식을 비교하였다. 표 4는 입력되는 10000의 컨텍스트 정보 중에 10개의 시나리오 컨텍스트 모두를 찾는데 비교되는 평균 비교회수 결과를 나타낸다.

<표 4> 평균 비교 회수 비교표



위의 <표4>를 보면, 하나의 대기 큐의 비교하는 비교횟수와 Type에 따른 Multi큐의 비교횟수간의 차이를 보여준다. 하나의 큐에서 비교할 경우에는 최상의 경우(Best_Case)와 (최악의 경우)Worst_Case간의 비교 횟수의 차이가 심하게 나는 반면, Multi 큐에서는 큰 차이를 보이지 않는다. 그리고 비교 횟수 면에서도 multi큐와 비교 했을 때 현저한 차이를 나타낸다.

더 많은 Type과 Value가 있다면 이러한 평균 비교 횟수의 격차는 더 벌어질 수 있다.

5. 결론 및 향후과제

본 논문에서 제시한 알고리즘의 경우에는 유비쿼터스 환경에서 uWDL 워크프로우 서비스를 사용자에게 빠르게 제공하기 위해 센싱 되어 들어오는 컨텍스트(Context)의 비교횟수를 줄이기 위한 방안으로 설계 되어 제시하고 있다.

이와같은 알고리즘은 단순히 비교 횟수를 줄이는 방안으로 설계 되었기 때문에 유비쿼터스 환경에 사용하기 위해서는 쓰이는 환경에 맞게 개선 사항을 고려해야 한다.

향후 연구 방향은 이러한 유비쿼터스 환경에서 컨텍스트 비교를 위해 multi큐의 사용이 미치는 영향, 과 실제 환경에 적용할 때 고려 되어야할 제반 사항에 대한 검증단계가 이루어 져야 할 것이다.

참고문헌

[1] Mark Weiser, "Some Computer Science Issues in Ubiquitous Computing", In Communication of the ACM, 36(7), pp.4-7, July 1993.

[2] 전태연, 한주현, 최재영 유비쿼터스 환경을 위한 워크플로우 기반의 시니리오 편집기의 설계 및 구현, 정보과학회, 2004.

[3] L. Ardisson, A.Di Leva, G. Petrone, M. Segnan, M. Sonnessa, "Adaptive Medical Workflow Management for a Context-Dependent Home Healthcare Assistance Service", Electronic Notes in Theoretical Cimputer Science, Elsevier, pp. 59-68, 2006.

[4] Joo Han, Y Cho, and J Choi, "Context-aware Work-flow Service Framework," LNCS 3983, 2006.

[5] D. Hollingsworth, "The Workflow Reference Model",Technical Report TC00-1003, Workflow Management Coalition, 1994.

[6] A.K. Dey, Providing Architectural Support for Building Context-Aware Applications", Ph.D. Thesis, Geogia Institute of Technology, 2000

[7] 최종선, 조용윤, 최재영 "동적인 사용자 서비스 요구를 지원하는 상황인지 워크플로우 시스템" 정보과학회 2007.