상황인지 서비스 제공을 위한 상황정보 비교 방법

예재형, 조용성, 최종선, 최재영
숭실대학교 컴퓨터학부
E-Mail: {yes, yongseoung.cho, jongsun.choi, choi}@ssu.ac.kr

A Comparison Method of Situational Information for Providing Context-Aware Services

JaeHyung Ye, Yongseong Cho, Jongsun Choi, Jaeyoung Choi
Dept. of Computer Science, Soong-Sil University

요 약

상황인지 기술은 사용자 주변의 정보를 이용하여 사용자의 현재 상황을 판단하는 기술로써 사용자의 요청 없이 최적화된 시점에 서비스를 제공하기 위한 것이다. 이러한 상황인지 기술은 일반적으로 별도의 추론 과정을 거쳐 응용 서비스에 적용된다. 본 논문에서는 서비스를 제공하기 위한 상황정보 비교 방법을 제안한다. 제안하는 방법은 상황인지 워크플로우 언어로 표현한 상황정보와 센서로부터 발생한 컨텍스트에 바탕을 둔 상황정보를 비교한다. 이를 통해 사용자 주변의 상황을 판단할 수 있으며 적합한 응용 서비스를 제공할 수 있다. 실험에서는 RDF 기반의 고수준 상황정보를 포함하고 있는 제약조건 테이블을 이용하여 상황정보의 비교 과정을 보인다.

KeyWords : Context-Aware, Situational information, Constraint Table, Context-Aware Workflow Language, Context. 상황인지, 상황정보 비교, 제약조건 테이블, 상황인지 워크플로우 언어, 컨텍스트.

1. 서론

상황인지 기술은 우리의 전반적인 일상생활부터 컴퓨팅 전 분야에서 적용되고 있다. 이러한 상황인지 기술들은 센서 데이터를 근본으로 하고 있으며 이를 처리하기 위해 별도의 추론 과정을 거치는 시스템들이 등장하고 있다[1-3]. 그들 중 하나는 워크플로우 기반의 상황인지 시스템으로, 특정 도메인에서 사용자에게 제공하기 위한 서비스들과 상황정보를 표현할 수 있어 상황인지 분야에서 주목받고 있다.

워크플로우는 특정 도메인에서 종속성이 있는 프로세스 단위의 작업들을 일반적인 규칙들에 의거하여 표현하는 기술이다[4]. 이를 구체적으로 표현하기 위하여 Workflow Management Coalition(WfMC)을 필두로 WSFL[5], XLANG[6], WSBPEL[7] 등과 같은 워크플로우 언어들이 등장하였고, 워크플로우 기반의 상황인지 시스템들은 워크플로우 언어들을 확장하여 상황정보[1]와 서비스를 정의하였다. 이러한 시스템들에서는 상황정보와 서비스 정의방법에 대한 특별한 표준이 없기 때문에 각각의 시스템들이 정의된 상황정보와 센서로부터 발생하는 컨텍스트[2]에 대하여 구체적인 비교 방법이 요구된다. 또한, 이러한 비교는 상황인지의 핵심이기 때문에 간결하고 합리적인 과정이어야 한다.

이에 본 논문에서는 센서로부터 발생한 컨텍스트에 바탕을 둔 상황정보와 Context-Aware Workflow Language (이하, CAWL)[9]로 표현한 상황정보를 비교하는 방법을 제안한다. 새롭게 제안하는 비교 방법은 비교에 필요한 제약조건들을 해쉬 테이블로 만들어 활용한다. 이를 설명하기 위하여 기존의 CAWL을 사용한 상황인지 시스템에서 시도하던 상황정보 비교 방법을 보이고, 새로운 상황정보 비교 방법으로 인한 비교 횟수와 데이터 생성량의 차이점을 보인다.

2. 관련연구

사용자의 현재 상황에 맞는 적합한 서비스를 제공하기 위해서는 컨텍스트와 상황정보를 비교하는 상황인지가 필요하다[10].

미국의 미네소타 대학에서는 상황인지 서비스 제공을 위한 Context-Aware Role Based Access Control 모델을 제시하고 있다[1]. 이 모델에서는 사용자를 역할 범주로 분류하여 서비스 목록들을 정의하고 사용자의 위치를 기

* 본 논문은 한국 산업통산자원부의 로봇산업융합핵심기술사업 프로그램 (No. 10048474)의 지원으로 수행되었습니다.

1) 상황정보: 본 논문에서의 "상황정보"는 고수준으로 표현된 상황정보인 RDF 기반으로 작성된 triplet의 의미로 사용하였다.

2) 컨텍스트: 본 논문에서의 "컨텍스트"는 온도, 습도, 위치, 시간 등의 저수준 상황정보의 의미로 사용하였다[8].

준으로 서비스를 제공한다. 따라서 사용자가 위치에 존재하지 않을 때에도 서비스를 제공받기 위한 예외처리가 필요하다. 또한, 제안하는 역할, 위치 등의 기술방법이 모델로만 제시되었기 때문에 구체적인 기술 방법이나 규약이 정의되어 제시될 필요가 있다.

ETRI에서 진행된 프로젝트 CAMUS(Context Middleware For URC Systems)에서는 상황인지를 위한 상황정보를 UDM(Universal Data Model)을 통해 표현하고 있다[2]. UDM은 노드를 통해 표현되는 트리 구조를 지니고 있으며 노드는 고유한 ID와 타입을 가진다. 또한, 이러한 노드들 중에는 스페셜 노드가 있어 값이 저장될 수 있다. 그리고 노드들은 Association을 통해 연결되며 Assocation은 방향성과 의미를 가진다. 이러한 UDM을 통해 상황정보를 쉽게 표현할 수 있으며 정의할 수 있다.

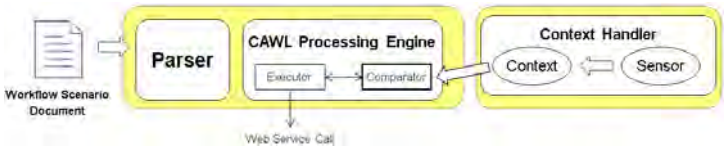
아일랜드의 코크 대학에서는 Data Management System-Context Architecture를 제안하였다[3]. 이 시스템에서는 사용자, 위치, 시간을 대상으로 다수의 센서가 동작하여 상황인지를 한다. 주목할 점은, 상황인지를 하기 위하여 정의된 상황정보의 시간 값을 참조하여 해당 시간에만 상황인지를 시도하여 비용을 절약한다는 것이다. 그러나 정의된 대다수의 상황정보의 규칙들의 시간 값이 길 경우를 고려하지 않았다.

중국 베이징의 북경이공대학에서는 Web Ontology Language 기반의 워크플로우 모델을 제시하였다[11]. 워크플로우로 표현되는 서비스들은 시작과 끝, 플로우의 반복과 분기 등이 온톨로지를 통해 표현되고 있다. 이 모델에서는 서비스 전이 조건을 위한 제약사항들이 온톨로지의 추론 기능을 통해 반영된다.

3. 상황정보 비교 방법

3.1 상황정보 비교를 위한 요구사항

본 논문에서 제안하는 상황인지 서비스 제공을 위한 상황정보 비교 방법을 위해 CAWL을 사용하는 기존의 상황인지 기반 워크플로우 처리 시스템을 제시한다.



[Figure 1] A System Architecture of the existing Workflow Handler

기존의 워크플로우 처리기는 상황인지를 통해 사용자에게 적합한 서비스를 제공하기 위한 시스템이다. [Figure 1]에서는 기존의 워크플로우 처리기의 구조를 보여주고 있다. 다음은 처리기의 구성 요소이다.

- **워크플로우 시나리오 문서** : CAWL을 사용하여 작성된 문서이다. 서비스들은 워크플로우 패턴이 적용되어 기술된다. CAWL에서 제공하는 변수들을 사용하여 서비스 제공을 위한 상황정보들(제약조건들)이 기술되어 있다.
- **파서** : 워크플로우 시나리오 문서를 대상으로 파싱을 진행한다. CAWL 처리 엔진이 수행하기 위해 필요한 데이터들을 적절한 형태의 지료구조로 제공한다.
- **CAWL 처리 엔진** : 실행기는 파서가 제공하는 자료구조를 참조하여 서비스 단위로 쓰레드를 만들어 운용한다. 비교기는 실행기로부터 정의된 상황정보를 전달받아 상황인지를 위한 상황정보 비교를 하고, 결과를 실행기에게 알려준다.
- **컨텍스트 핸들러** : 저수준의 컨텍스트를 고수준의 상황정보로 가공한다. 여기서 저수준의 컨텍스트는 센서로부터 얻을 수 있는 이진 값이며, 고수준의 상황정보는 이진 값을 바탕으로 가공된 RDF[12] 기반의 triplet을 의미한다.

컨텍스트 핸들러는 가용한 모든 센서를 동원하여 컨텍스트 정보를 고수준의 상황정보로 만들어낸다. 가용한 센서로 만들어 낼 수 있는 모든 경우의 수에 해당하는 RDF 기반의 triplet을 실시간으로 만들어 내며 이들 모두를 CAWL 처리 엔진의 비교기에 전달한다. CAWL 처리 엔진은 상황인지를 위한 제약조건을 바탕으로 컨텍스트 핸들러로부터 전달받은 다수의 고수준 상황정보를 순차적으로 비교함으로써 상황인지를 수행한다.

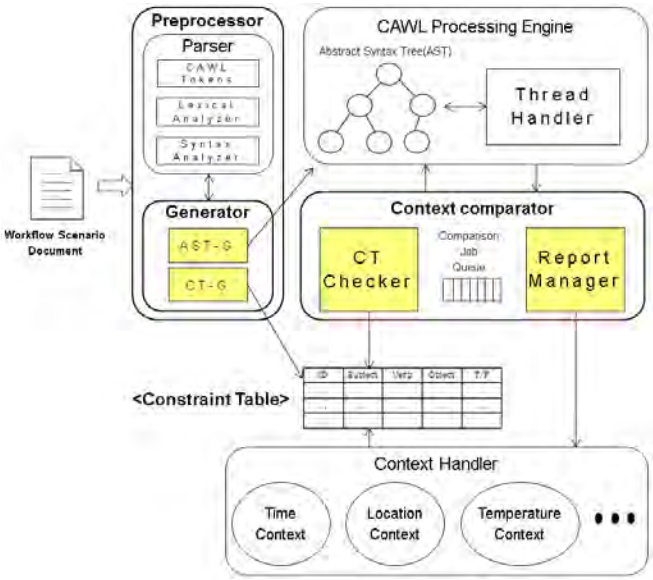
3.2 개선된 비교 방법을 위한 변동사항

본 논문에서 제안하는 상황정보 비교방법은 기존의 비교 방법의 문제점을 개선하기 위하여 특별한 자료구조로 형성된 데이터들을 이용한다. 이러한 자료구조를 만들기 위해 전처리기에 생성기가 추가되었다. 또한, 기존의 CAWL 처리 엔진의 비교기를 제거하고 새로운 컨텍스트 비교기를 만들어 사용하였다.

전처리기는 상황정보 비교를 위해 필요한 파싱이나 자료구조 생성을 담당한다. [Figure 2]에 제시된 전처리기는 기존의 워크플로우 처리기에는 없었던 것으로, 파서와 생성기를 포함하고 있다. 파서는 기본적으로 워크플로우 시나리오 문서에 대한 파싱을 수행하고, 생성기는 파서에서 발생된 이벤트를 감지하여 수행된다.

생성기는 컨텍스트 비교기나 CAWL 처리 엔진에서 필요로 하는 데이터들을 적절한 자료구조로 만들어 주며 두 개의 세부 생성기로 구성되어 있다. 하나는 추상 구문 트리 생성기(Abstract Syntax Tree Generator, 이하 AST-G)이며 다른 하나는 제약조건 테이블 생성기(Constraint Table Generator, 이하 CT-G) 이다.

생성기의 AST-G는 워크플로우 시나리오 문서의 <node> 태그들을 대상으로 AST를 구성한다. <node> 태그에는 서비스 제공을 위한 제약조건들이 triplet 형태로 기술되어 있으며 웹 서비스 호출을 위한 메소드 정보도



[Figure 2] A System Architecture of the suggested Workflow Handler

기술되어 있다.

생성기의 CT-G는 워크플로우 시나리오 문서 안에 존재하는 모든 제약조건들을 제약조건 테이블에 등록하여 만들어 낸다. 제약조건 테이블은 유일한 Key를 반드시 1개 가지는 해싱 기법이 적용된 테이블이다. 이러한 제약조건 테이블은 컨텍스트 핸들러와 컨텍스트 비교기에서 함께 참조할 수 있다. 또한, 테이블에 등록될 제약조건들은 유일한 키(ID)값이 부여되어 함께 보관된다. 이러한 ID 값은 AST에도 반영된다. 만약 같은 triplet이 제약조건 테이블에 존재한다면 해당 triplet의 ID를 AST에 반영하기만 하면 된다.

컨텍스트 비교기는 기존의 워크플로우 처리기에서 벗어나 독립적으로 수행되며 기능이 확장되었다. 컨텍스트 비교기에는 CAWL 처리 엔진으로부터 요청받은 쓰레드 정보, 제약조건의 ID들이 보관되는 Comparison Job Queue가 있다. 그리고 CAWL 처리 엔진과 컨텍스트 비교기, 컨텍스트 비교기와 컨텍스트 핸들러간의 통신임무를 담당하는 Report Manager가 있다. 또한, 컨텍스트 핸들러에 의해 발생된 이벤트를 감지하여 수행되는 제약조건 테이블 확인자(Context Table Checker, 이하 CT Checker) 세 가지로 구성되어 있다.

컨텍스트 비교기의 Report Manager는 CAWL 처리 엔진 또는 컨텍스트 핸들러 측과 통신하는 일을 담당한다. 예를 들어 CAWL 처리 엔진의 쓰레드가 상황인지가 필요할 경우, 쓰레드 핸들러를 통해 보내오는 ID 값을 수신한다. 이 때 ID 값이 2개 이상일 경우, 제약조건간의 논리연산 정보도 함께 전송받게 된다. 그리고 전송받은 정보를 Comparison Job Queue에 보관한 후에 컨텍스트 핸들러 측에 ID를 전송한다. ID 값을 전송받은 컨텍스트 핸들러는 제약조건 테이블을 참조하여 현재 필요한 센서가 무엇인지를 한정 지을 수 있다. 이를 통해 컨텍스트 핸들러는 한정된 센서 정보를 고수준의 데이터로 가공하여 이 정보를 Table에 갱신시키고 이벤트를 발생시킨다.

컨텍스트 비교기의 CT Checker는 컨텍스트 핸들러가 테이블을 갱신시키면서 발생시킨 이벤트를 감지하여 수행한다. 이벤트가 감지되면 Comparison Job Queue의 대기중인 상황정보 요청들을 대상으로 제약조건 테이블을 참조하고 제약조건간의 논리연산을 수행한다. 수행한 결과가 참이면 Report Manager에게 통보하여 CAWL 처리 엔진의 쓰레드 핸들러에게 알려주고 Constraint Job Queue에서 해당 상황정보 요청을 삭제한다.

4. 실험 및 평가

본 논문에서는 상황인지 워크플로우 언어를 기반으로 작성된 시나리오 문서를 통해 상황정보를 비교하기 위한 방법을 소개하였다. [Table 1]은 CAWL을 기반으로 작성된 워크플로우 문서에서의 제약조건들의 예시이다.

[Table 1] Example of Constraint described in CAWL Document

```
<CAWL name="" version="1.01" targetNamespace="">
  ...
  <node name="MusicTeaPreparation">
     <constraint>
       <subject type="Robot">
?var4MTRoomService/robot1</subject>
       <verb>isLocated</verb>
       <object type="Room">
?=var4MTRoomService/meetingRoomNum
       </object>
     </constraint>
  </node>
  <node name="DisplayLightInfo">
     <constraint>
       <subject type="militaryTime">
?var4TimeInfo/rsvTime
       </subject>
       <verb>is</verb>
       <object type="militaryTime">
          crtTime
       </object>
     </constraint>
  </node>
  ...
```

전처리기의 CT-G는 [Table 1]의 시나리오 문서를 대상으로 제약조건 테이블을 [Table 2]와 같이 생성한다. [Table 2]는 앞서 언급했듯이 유일한 ID를 키 값으로 가지는 해시 테이블이다.

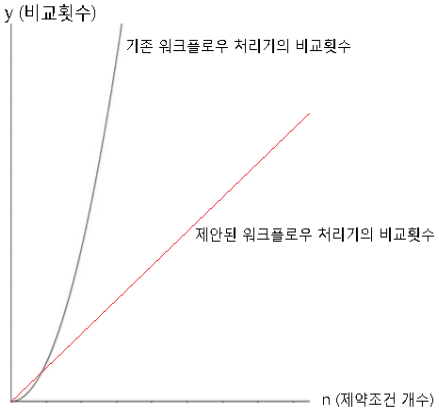
[Table 2] Constraint Table

ID	Subject	Verb	Object
0	R-307	isLocated	313
1	1205	is	crtTime
2	...	...	...
3	...	...	...

컨텍스트 핸들러는 컨텍스트 비교기로부터 특정 ID 값을 지니는 제약조건에 대한 요청이 있을 때에만 그에 해당하는 필요한 센서데이터만을 적시에 가공하면 된다. 이것은 컨텍스트 테이블에 유일한 ID 값이 부여된 제약조건들을 유지하기 때문에 가능하며 컨텍스트 핸들러는 가용

한 모든 센서 데이터를 triplet으로 가공하는 일을 실시간으로 할 필요가 없다.

[Figure 3]에서는 기존의 비교횟수와 개선된 비교횟수를 그래프 형태로 보여준다.



[Figure 3] The number of comparisons between the existing workflow handler and the suggested workflow handler

기존 시스템에서의 비교 횟수는 컨텍스트 핸들러가 센서를 통해 만들어 낼 수 있는 모든 경우의 고수준의 상황정보를 n개라 가정하고, 시나리오 문서상에 존재하는 모든 제약조건의 개수들도 n개라 가정하면, n^2 번의 비교가 필요하다. 따라서 그래프 상에서 비교 횟수인 y값에 대하여 2차함수 형태로 증가하는 것을 볼 수 있다.

새롭게 개선된 시스템에서의 비교 횟수는 컨텍스트 핸들러가 센서를 통해 만들어 낼 수 있는 모든 경우의 고수준의 상황 정보가 n개라 하더라도 필요한 시점에 필요한 상황정보만을 가공한다. 그러므로 시나리오 문서상에 존재하는 모든 제약조건의 개수가 n개라 하면, n 번의 비교만이 필요하다. 또한, 제약조건 테이블은 해쉬테이블로 구성되어 있기 때문에 원하는 제약조건을 추출할 때 필요한 시간복잡도는 상수이다. 따라서 그래프 상에서 개선된 비교횟수가 1차함수의 형태로 증가하는 것을 볼 수 있다.

5. 결론

본 논문에서는 상황인지 서비스 제공을 위한 상황정보 비교방법을 제안했다.

본 논문의 컨텍스트 비교기는 유일한 ID 값이 부여된 제약조건들을 유지하는 제약조건 테이블을 사용하였다. 이를 통해 상황인지가 필요한 시기에만 적은 비교 비용으로 상황인지를 수행하였으며 정의된 제약조건들의 중복이 제거되었다. 그리고 개선된 시스템에서 전처리기를 제안하여 객체마다 기능성을 강조하였고 CAWL 처리 엔진과 컨텍스트 비교기를 분리하여 각 모듈의 특성을 부각시켰다. 이와 같은 상황정보 비교 비용의 간소화 노력은 차후에 하나의 워크플로우 처리기 시스템에서 다수의 워크플로우를 제어하게 될 경우에도 큰 이점이 될 수 있다. 이를 위해서는 다수의 워크플로우가 제어되는 구체적인 워크플로우 시나리오 문서의 기술 방법의 정의가 필요하며, 각종 공유자원의 동기화 문제를 고민해보고 해결해야 한다.

참고문헌

[1] D. Kulkarni, A. Tripathi, "Context-aware role-based access control in pervasive computing systems", *SACMAT : Proceedings of the 13th ACM symposium on Access control models and technologies*, pp. 113-122, 2008.

[2] H. Kim, M. K. Kim, K. W. Lee, Y. H. Suh, J. M. cho, Y. J. Cho, "Context-Aware Server Framework for Network-based Service Robots", *SICE-ICASE* International Joint Conference, pp. 18-21, 2006.

[3] J. Herbert, J O'Donoghue, X. Chen, "A context-sensitive Rule-based architecture for a smart building environment.", *Future Generation Communication and Networking, FGCN*: Second International Conference on, vol. 2, pp. 437-440, 2008.

[4] Zhu, Yajie, Tai Xin, and Indrakshi Ray. "Recovering from malicious attacks in workflow systems." Database and Expert Systems Applications. Springer Berlin Heidelberg, pp. 14-23, 2005.

[5] Leymann, Frank, "Web services flow language (WSFL 1.0).", (2001).

[6] Thatte, Satish, "XLANG: web services for business process design, 2001." Microsoft http://www. gotdotnet. com/team/xml-wspecs/xlang-cl/default. htm (2001).

[7] Weerawarana, Sanjiva, et al. "Web services platform architecture: SOAP, WSDL, WS-policy, WS-addressing, WS-BPEL, WS-reliable messaging and more.", Prentice Hall PTR, 2005.

[8] Chen, Guanling, David Kotz. "A Survey of Context-Aware Mobile Computing Research. *Technical Report* TR2000-381, vol. 1, no. 2, 2000.

[9] J. S. Choi, Y. Y. Jo, J. Y. Choi, "The Design of a Context-Aware Workflow Language for Supporting Multiple Workflows", *Journal of Korean Society for Internet Information* 11(1), pp. 145-158, 2009.

[10] J. H. Han, Y. Y. Cho, E. H. Kim, J. Y. Choi, "A Ubiquitous Workflow Service Framework," *ICCSA06* pp.30-39, 2006.

[11] Wang, Pengfei, Huifang Li, and Baihai Zhang. "Context-aware workflow modeling approach using OWL." *Control and Decision Conference : The 26th Chinese.* IEEE, pp. 4161-4165, 2014.

[12] F. Manola and E. Miller, "RDF Primer", W3C Recommendation, 2004.