상황인식 처리를 위한 상황정의 스크립트 언어의 설계

태봉섭[O], 장재우
전북대학교 컴퓨터공학과
{bstae, jwchang}@dblab.chonbuk.ac.kr

김정기, 신창민
한국전자통신 연구원
{jkk, cmshin}@etri.re.kr

Design of Context Definition Script Language for Context Awareness Processing

Bong-Sub Tae[O] Jae-Woo Chang
Dept. of Computer Engineering,
Chonbuk National University

Jeong-Ki Kim, Chang-Min Shin
Electronics and Telecommunications
Research Institute

요 약

최근 언제 어디서나 네트워크에 접속할 수 있고, 모든 사물이 네트워크로 연결되어 있는 편재형(Ubiquitous) 컴퓨팅에 관한 연구가 활발히 진행 중이다. 편재형 컴퓨팅을 위해서는 사용자 주변의 상황을 감지하는 상황인식(Context Awareness)과 감지된 상황에 대해서 알맞은 서비스를 하기 위한 미들웨어가 필요하다. 또한, 이러한 지능적인 서비스를 하기 위해 상황을 편리하게 표현할 수 있는 체계적인 언어가 필요하나, 이러한 언어에 대한 연구는 미흡한 실정이다. 따라서 본 논문에서는 이러한 상황 인식 처리를 위한 상황 정의 스크립트 언어에 대해서 설계한다.

1. 서 론

최근 하드웨어 기술 발전에 힘입어 PDA, 스마트폰, 인터넷 TV, 차세대 게임기 등이 속속 출현하고 있다. 아울러, 컴퓨터와 통신 기술(Bluetooth, IEEE 1394, Power Cable Network, IEEE 802.11b 등)은 수많은 다른 분야의 기술들과 접목되어 일상생활에 활용이 가능한 컴퓨팅 환경을 제공하고 있는 추세이다. 제록스 PARC의 마크 와이저(Mark Weiser)는 사람을 포함한 현실 공간에 존재하는 모든 대상물을 기능적, 공간적으로 연결해 사용자에게 필요한 정보나 서비스를 즉시에 제공할 수 있는 컴퓨팅 환경을 편재형(유비쿼터스) 컴퓨팅[1,2]이라고 하였다.

다양한 형태의 컴퓨터들이 사용자가 인식하지 못하는 형태로 현실 공간의 사물과 환경 속으로 스며들어 사용자가 필요로 하는 서비스를 제공하기 위해서는, 일상생활 곳곳에 편재된 센서 및 컴퓨터들이 수집한 각종 정보를 효과적으로 상호 공유하여 사용자 및 주변 환경의 컨텍스트를 알아내는 상황인식 처리 기술이 요구된다. 이러한 상황인식 처리 기술은 사용자를 중심으로 하는 주변 환경과 사용자간 혹은 사용자와 장치간의 상호 운용성을 자동적으로 선택하여 지원해 줌으로써, 편재형 컴퓨팅 환경에서 가장 중요한 핵심 기술로 발전되고 있다. 한편, 소프트웨어 개발을 위해서 개개의 특성을 가진 다양한 프로그래밍 언어가 개발되었지만, 주어진 상황(Context)을 정의(definition)할 수 있는 상황인식을 위한 언어는 아직까지 개발이 미흡한 상황이다. 이를 위해, 본 논문에서는 주어진 상황 및 상황에 따르는 행동(action)을 효과적으로 기술하여 감지된 상황에 대해서 지능적으로 서비스를 제공할 수 있는 상황인식 처리를 위한 상황정의 스크립트 언어를 설계한다.

본 논문의 구성은 다음과 같다. 제 2 장에서는 미들웨어에 대해서 기술하고, 제 3 장에서는 상황인식 처리를 위한 컨텍스트 정의 언어에 대해서 기술한다. 제 4 장에서는 결론 및 향후 연구를 제시한다.

2. 상황인식 처리를 위한 미들웨어

미들웨어는 IRISA/INRIA[3]의 미들웨어 모델을 참조하여 설계한 것으로 상황정의 스크립트 처리기를 부가적으로 추가하여 범용성을 높였다. 각각의 구성 요소를 살펴보면, 그림 1과 같이 3단계의 계층으로 구성되며, 이는 탐지 및 모니터링 계층, 상황인식 계층, 응용계층이다.

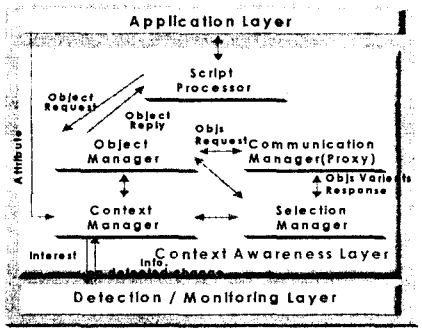

그림 1. 상황인식 처리를 위한 미들웨어

탐지 및 모니터링 계층은 원격 객체의 위치정보나 응용 프로그램과 상호작용에 따른 이벤트, 그리고 CPU 사용량, 메모리의 사용량과 같은 컴퓨팅 자원이나 네트워크 상태를 모니터링 할 수 있는 기능을 담당하며, 이 계층에서는 응응 시스템에 따라서 탐지 모듈의 유연한 형태를 가진다. 상황인식(context awareness) 계층은 편재형 컴퓨팅을 위한 상황인식 처리 기술의 핵심 계층으로서 미들웨어 역할을 담당한다. 이 계층은 크게 스크립트 처리기, 원격 객체 관리자, 컨텍스트 관리자, 컨텍스트 선택 관리자, 통신 관리자의 5가지의 관리자로 구성되고, 상황정의 스크립트 언어 처리기를 위한 API로서 제공된다. 스크립트 처리기는 응용 프로그램의 상황인식 정의 스크립트의 각 내용을 분석하여 규약에 명시된 역할을 수행하도록 한다. 원격 객체 관리자는 각 응용 프로그램에서 현재 사용하고 있는 원격 객체(컨텍스트 객체)에 포함되어 있는 모든 정보를 위한 데이터 구조를 관리한다. 컨텍스트 관리자는 사용자의 취향, 이동 단말 기기의 성능, 현재 위치정보 등을 포함하고 있는 주어진 환경이나 상황에 대한 컨텍스트 정보를 관리한다. 컨텍스트 선택 관리자는 응용 프로그램을 통해 사용자에게 전송될 조건에 부합되는 컨텍스트 정보를 선별한다. 통신(프락시) 관리자는 컨텍스트 서버와 통신을 담당하며, 예상하지 못한 네트워크의 장애로 인한 문제에 대비한다. 응용 계층은 하부의 상황인식 계층인 미들웨어와 독립적으로 동작하면서 미들웨어의 응용 프로그래밍 인터페이스(API)를 통해 상황인식 처리 기술에 기반을 둔 다양한 응용 프로그램을 개발하기 위한 기능을 수행한다.

3. 상황인식을 위한 상황정의 스크립트 언어의 설계

상황정의 언어는 주변의 환경에 영향을 미치는 컨텍스트 객체와의 상호 작용 속에서 어떠한 상황에 대한 조건(condition)을 정의하고 그 조건에 상황이 일치하면 적절한 행동(action)을 취하게 하는 구조로서 범용적으로 사용할 수 있으며, 상황인식 처리를 위해 무수히 반복되는 코드를 스크립트 언어를 통해서 간략하게 만들 수 있다.

3.1. 컨텍스트 객체 정의

컨텍스트 객체는 상황인식 소프트웨어 주위의 모든 사물이나 장치를 나타내는 논리적인 개념이다. 객체는 다수개의 속성값으로서 이루어져 있으며, 속성 값들 중 일부는 하드웨어적인 장치에 의해서 갱신된다. 속성 값들은 뒤에서 언급할 컨텍스트의 조건절(condition clause)에서 사용된다. 컨텍스트 객체의 정의 용법은 다음과 같다.

```
ContextObject object_name (
        data_type element_name1,
        data_type element_name2,
        data_type element_name3
                    :
                    . :
)
```

object_name과 element_name은 일반적인 프로그래밍 언어의 변수 명명 규칙과 같다. data_type은 문자와 문자열 데이터 타입으로서 string이 있고, 수치 데이터 타입으로서 int, float, long, double, time, date가 있다.

3.2. 컨텍스트의 정의/삭제

컨텍스트의 정의는 어떠한 조건(condition)의 상황이 발생하면 어떠한 동작(action)을 수행함을 정의하는 것으로써 조건절과 그 조건절이 만족되었을 때 수행하는 동작을 나열한다. 여기서는 어떠한 컨텍스트의 틀(template)을 정의하는 것으로 컨텍스트 인스턴스가 추가될 때 동작이 가능한 형태가 된다. 사용법은 다음과 같다.

```
ContextDefine context_name
Condition     (condition_expression)
Action        function_name
              [before/when/after time_value] or
              [from time_value to time_value]
Param         (param1, param2, ... )
```

컨텍스트를 정의할 때 ContextDefine이 사용되며, 부가적으로 Condition, Action, Param 절이 함께 사용된다. 사용법을 살펴보면 ContextDefine에 의해서 컨텍스트가 정의되며, Condition을 통해서 해당하는 조건을 정의하고, Action과 Param에 의해서 수행해야 할 함수가 호출되어 매개변수가 전달된다. 여기에서 ContextDefine의 context_name은 컨텍스트의 정의된 이름이다. 각 절을 살펴보면, 첫째, Condtion의 condition_expression은 조건절로서 내부에서 괄호를 중첩해서 사용할 수 있고, 조건을 표현하기 위한 관계 연산자로서 등호(=)와 부등호(<, >)를 사용하고, 논리연산을 위해서 논리연산자인 AND, OR, NOT을 사용한다. 그 밖의 연산자로서 [during time_value]이 있는데, 만약 during time_value 연산자가 추가되면 조건 절이 time_value의 시간만큼 지속되어야 참이 된다. 여기서 time_value는 시간의 값으로서 표현하고, 수치 뒤에 s,m,h 가 추가됨에 따라 각각 초, 분, 시간으로 구분된다. 둘째, Action절의 function_name/NULL과 [while/when/after time-value] or [from time_value to time_value]에 대해 설명한다. function_name은 조건이 만족되었을 경우 실행하는 함수의 이름이 기술된다. 호출하는 함수는 용도에 따라 미리 만들어져 준비된 것일 수도 있고, 사용자의 필요에 따라 새로 만들어서 사용하는 경우도 있다. 만약 함수를 호출할 필요가 없을 경우에는 function_name에 NULL을 사용한다. function_name 다음의 시간에 대한 옵션항은 함수를 수행함에 있어 시간에 따라 어떻게 실행할 것인지를 결정하며, 옵션 값으로서 생략이 가능하다. 어떠한 시점(when)을 기준으로 before는 그 시점 이전에 한번만 Action을 수행하고, after는 그 시점 이후에 한번만 Action을 수행하는 것이다. from, to는 어떤 정해진 시간동안 계속적으로 실행하는 것으로 그림 2에서 도식화 하여 보여준다.

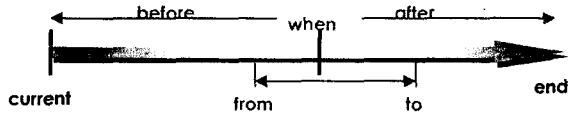

그림 2. Action에 대한 시간과의 관계

마지막으로 셋째, Param (param1, param2, ...)은 Action에 의해서 실행되는 함수에 전달할 매개변수의 리스트이다. 만약 전달할 매개변수가 없을 경우에는 Param NULL로 표현할 수 있다.

앞에서 정의한 컨텍스트를 삭제 할 경우는 다음과 같다.

```
ContextDelete context_name
```

3.3. 컨텍스트 인스턴스의 삽입/삭제

컨텍스트가 정의되어 컨텍스트 인스턴스를 담을 수 있는 틀이 만들어졌다면, 컨텍스트에 내용을 담아 구체화하는 것이 컨텍스트 인스턴스이다. 사용법은 다음과 같다.

```
ContextInsert instance_name into context_name
Condition     (condition_value1, condition_value2, ... )
Param         (param_value1, param_value1, ... )
```

컨텍스트에 컨텍스트 인스턴스를 추가 할 때 ContextInsert, Condition, Param의 절로 표현한다. ContextInsert의 context_name에는 인스턴스가 삽입될 컨텍스트의 이름을 기술하며, instance_name에는 컨텍스트 인스턴스의 이름을 기술한다. Condition절과 Param절의 각 value에는 리터럴 즉, 상수가 기술된다. 컨텍스트 정의할 때 condition_expression의 조건 표현부에서 컨텍스트 오브젝트의 속성값과 '변수' 가 비교되는 경우가 있는데, 이 '변수' 에 해당하는 실제 값이 컨텍스트 인스턴스 삽입시 Condition절에 사용되는 각 condition_value이다. 같은 맥락으로 Param절의 parma_value의 각 값은 컨텍스트 정의시 Action에서 호출되는 함수의 Param에 해당되는 변수의 실제 상수값을 담고 있다. 컨텍스트 인스턴스의 삭제는 다음과 같다.

```
ContextDelete instance_name in context_name
```

3.4. 컨텍스트 인스턴스의 시작/정지 및 주석표현

컨텍스트의 정의와 컨텍스트 인스턴스의 삽입이 완료되면 실제 그 인스턴스를 활성화하여 계속적으로 조건의 상태를 판단하고 해당되는 행동을 취하도록 하는 것이 컨텍스트 인스턴스의 시작이다. 반대로 컨텍스트 인스턴스의 상황 판단을 멈추도록 하는 것으로서 사용법은 다음과 같다

```
ContextStart instance_name in context_name
ContextStop instance_name in context_name
```

어떤 컨텍스트의 모든 인스턴스를 시작하거나 정지할 경우에는 instance_name에 별표(* - asterisk)를 사용한다. 스크립트 기술시 주석 표현은 다음과 같다. 두개의 겹사선(double slash)을 사용함으로써 주석 표현을 한다

```
// annotaion_statement
```

3.5. 상황정의 스크립트 언어의 예제

여기서는 앞에서 정의된 각 용법을 이용해 간단한 예제를 소개한다. 예를 들어, 각 사람이 선호하는 음악이 있고 이 사람들이 각 방을 이동하는데, 만약 한 사람이 방에서 2초이상 대기하게 되면, 1초 후에 정해진 음악을 재생한다. 이는 다음과 같다.

```
ContextObject person (
string name,
string location
)
```

```
ContextDefine music
Condition (((person.name = someone_name) AND
((person.location = someone_location)during 2s))
Action PlayMusic from 1s to end
Param (music_name)
```

이제 컨텍스트 객체와 컨텍스트가 정의되었다. 이때 "김철수"라는 사람이 "7401" 호의 방에 있을 때는 "아리랑"의 음악을 재생하고, "이영희"라는 사람이 "7429" 호의 방에 있을 때는 "도라지"의 음악을 재생할 경우는 다음과 같다.

```
ContextInsert kim into music
Condition ("김철수", "7401")
Param ("아리랑")

ContextInsert lee into music
Condition ("이영희", "7429")
Param ("도라지")
```

여기서 someone_name, someone_location, music_name이 인스턴스에 의해서 구체화 되었다. 이 인스턴스의 각 조건을 주기적으로 체크하도록 하기 위해서는 다음과 같은 ContextStart가 필요하다. 여기서 ContextStart 구문 사용 시 별표(*)를 사용함으로써 music의 모든 인스턴스의 조건이 체크되도록 하였다.

```
ContextStart * in music
```

3.6 상황정의 스크립트 언어의 구문 구조

스크립트 언어의 구문구조는 ContextDefine을 예로 들어 개념적으로 설명을 한다. ContextDefine을 위한 각 예약어와 각 구문 구조 요소의 정규 표현식을 살펴보면 표 1, 2와 같다.

표 1. 스크립트의 예약어

종 류	심 볼	역 할	예약어
명령어	예약어와 같음	스크립트 명령어	ContextDefine,
부가 명령어	예약어와 같음	스크립트 명령어를 위한 추가적인 명령어	Condition, Action, Param, in, into
시간 명령어	예약어와 같음	스크립트 명령어를 위한 시간 관련 추가 명령어	during, when, while, after
관계 연산자	re_op	대소 비교를 위한 연산자	=, <>, <,>, <=, =>
논리 연산자	lo_op	논리 연산을 위한 연산자	and, or , not, AND, OR, NOT

표 2. 각종 변수명과 수치 데이터들의 정규 표현식

종 류	심 볼	정규 표현식
변수 이름	var	[a-zA-Z][a-zA-Z0-9]*
컨텍스트 이름	context_name	
컨텍스트 오브젝트 이름	context_obj_name	
함수 이름	func_name	

표에서 정의된 각 심볼을 이용하여 컨텍스트를 정의하는 ContextDefine의 구문구조는 다음과 같다

```
ContextDefine context_name
Condition condition_expression
Action func_name act_time_option
Param ( func_name (,func_name)* )
```

위의 구문구조에서 필요한 각 요소를 나열하면 다음과 같다.

```
condition_expression : (condition time_option (lo_op condition
cond_time_option)* )
```

```
cond_time_option : (during time_value) {0,1 }
```

```
condition :
context_name.context_obj_name re_op var |
context_name.context_obj_name re_op
context_name.context_obj_name |
var re_op var |
var re_op context_name.context_obj_name
```

```
act_time_option :
when time_value | before time_value | after time_value |
from time_value to time_value
```

3.7 상황정의 스크립트 언어 처리기

상황정의 스크립트 언어 처리기는 정의된 구문에 따라 생성된 스크립트 언어 파일을 문법 분석 및 구문 분석을 통해 처리하는 인터프리터 프로그램이다. 전체적인 구조는 그림 3과 같다.

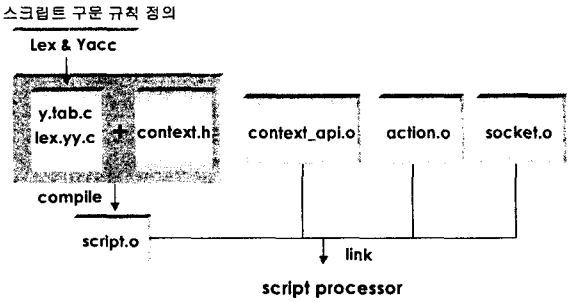

그림 3. 상황정의 스크립트 처리기의 전체적인 구조

각각의 모듈을 살펴보면, script.o는 Lex와 Yacc를 통해 상황정의 스크립트 언어의 문법 및 구문 규칙을 정의하여 컨텍스트 API를 호출하는 내용이 내재되어 있다. context_api.o에는 스크립트에서 기술된 모든 표현을 처리하는 API가 기술되어 있고, action.o에는 어떠한 조건을 만족했을 때 수행하는 함수들 즉, 컨텍스트 정의시 Action절에서 호출하는 함수들의 실제 내용이 기술된다. 마지막으로 socket.o에는 컨텍스트 객체와 통신을 위한 소켓 관련 API가 포함된다. 앞에서 언급한 모듈이 모두 링크되어 하나의 실행파일이 되어 상황정의 스크립트를 처리하는 프로그램으로서 동작하게 된다.

4. 결론 및 향후 연구

본 논문에서는 편재형 컴퓨팅을 위해 사용자 주변의 상황을 감지하는 상황인식을 위한 지능적인 서비스를 하기 위해 상황을 편리하게 표현 할 수 있는 언어인 상황 정의 스크립트 언어에 대해서 설계하였다. 향후 연구로는 상황정의 스크립트 언어를 바탕으로 상황 정보를 설계된 스크립트 언어의 실제 응용에 적용하여 사용하고, 이를 통해 상황인식 기반 응용 시스템을 구축하는 것이다.

[참고문헌]
[1] http://www.ubiq.com/hypertext/weiser/UbiHome.html, M. Weiser, "Ubiquitous Computing"
[2] M. Weiser, "Some Computer Science Issues in Ubiquitous Computing," Commnunications of the ACM, Vol 36(7), pp. 74-84, 1993
[3] P. Couderc, A.-M. Kermarrec, "Improving Level of Service for Mobile Users Using Context-Awareness," 18th *IEEE Symposium on Reliable Distributed Systems*, pp. 24-33, 1999.