자율적 커뮤니티 컴퓨팅 기반 유비쿼터스 시스템 플랫폼을 위한 효과적인 모니터링 방법

(An Efficient Monitoring Method for Ubiquitous System Platform Based on Autonomic Community Computing)

권 성 현 † 이 동 욱 †
(Sung-Hyun Kwon) (Dong-Wook Lee)

김 재 훈 †† 조 위 덕 †††
(Jai-Hoon Kim) (We-Duke Cho)

요 약 유비쿼터스 지능 공간은 사용자의 요구를 서비스하기 위하여 여러 종류의 스마트 객체들이 다양한 형태로 연결되어 있다. 따라서 사용자의 서비스 요구에 따라 서비스 구성 중에 과도한 부하가 발생할 뿐만 아니라 서비스 객체들 간의 자원 충돌이 발생할 수 있다. 결국, 각 서비스 객체들 간에 오작동이 발생해 사용자에게 잘못된 서비스를 제공하거나 서비스에 응답을 못 할 수도 있다. 이런 문제를 해결하기 위해서는 유비쿼터스 지능공간은 모니터링 시스템에 의해 다양한 스마트 객체들의 기능, 성능 및 상태를 모니터링 한다. 또한 로그를 기록하고 기록된 로그를 분석하여 사용자에게 지능 공간 서비스에 대한 최적화된 기능, 성능 및 서비스 품질을 개선 할 수 있어야 한다. 본 논문에서는 다양한 서비스 요구 패턴 등에 따른 성능 최적화와 시스템의 중단 및 오작동이 발생했을 때 자가 복구를 수행할 수 있는 모니터링 시스템을 제안한다.

키워드 : 유비쿼터스 지능 공간, 모니터링 시스템, 성능, 자가복구

Abstract Ubiquitous Smart Space connects many kinds of smart object to satisfy user's demands. Thus, not only excessive loads but also resource collisions occur among service objects while configurating services to fulfill the user's request. After all, as malfunction occurs between service objects, it offers wrong service to user or does not respond to service request. To solve these problems, Ubiquitous Smart Space is observed by monitoring system of function, performance, and status of objects. Therefore, optimized function, performance and quality of service of smart space service for user should be improved by recording log and analyzing recorded the log. In this paper, we suggest a novel monitoring system to optimize performance according to pattern of diverse services and execute self-recovery on system down and malfunction.

Key words : Ubiquitous Smart Space, Monitoring system, Performance, Self-recovery

1. 서 론

자율적 커뮤니티 컴퓨팅 기반 유비쿼터스 시스템 USPACC(Ubiquitous System Platform based on Autonomic Community Computing)은 동적으로 구성되고 이질적인 장비끼리 묶이는 유비쿼터스 환경에서 시스템 스스로가 환경의 변화에 따라 현재의 구성을 효과적으로 재구성 한다[1]. 그리고 상황 추론을 통해 사용자에 최적화된 지능 공간을 구성함으로써 사용자에게 편리하고 안전하며 쾌적한 환경을 제공한다. 하지만 USPACC는 사용자의 서비스 요구에 동적으로 서비스 객체를 구성하는 중에 과도한 부하가 발생할 뿐만 아니라 스마트 객체들 간의 자원 충돌이 발생 할 수 있다. 지능형 공간에서 서비스의 기능 및 성능 보장과 함께 신뢰도를 높이기 위해서는 객체들의 기능과 성능이 정상적으로 동작하는지 주기적으로 감시하는 관리 서비스를 제공해야 한다.

본 논문에서 제안하는 모니터링 시스템은 기본 서비스 단위의 기능을 하는 서비스 모듈(Service Module)과 서비스 모듈을 그룹으로 구성해 사용자에게 제공되는 최적화된 지능 공간을 효과적으로 모니터링하는 시스템이다. 이 모니터링 시스템은 각 서비스 모듈마다 기본동

· 본 연구는 21세기 프론티어 연구개발사업의 일환으로 추진되고 있는 정보통신부의 유비쿼터스 컴퓨팅 및 네트워크원천기반기술 개발사업의 지원에 의한 것임
· 이 논문은 2007 한국컴퓨터종합학술대회에서 '자율적 커뮤니티 컴퓨팅 기반 유비쿼터스 시스템 플랫폼을 위한 효과적인 모니터링 방법'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 아주대학교 정보통신전문대학원
liebeym@ajou.ac.kr
dwlee@ajou.ac.kr
†† 정 회 원 : 아주대학교 정보통신전문대학원 교수
jaikim@ajou.ac.kr
††† 종신회원 : 아주대학교 전자공학과 교수
chowd@ajou.ac.kr

논문접수 : 2007년 9월 28일
심사완료 : 2007년 12월 28일

Copyright@2008 한국정보과학회 : 개인 목적이나 교육 목적인 경우, 이 저작물의 전체 또는 일부에 대한 복사본 혹은 디지털 사본의 제작을 허가합니다. 이 때, 사본은 상업적 수단으로 사용할 수 없으며 첫 페이지에 본 문구와 출처를 반드시 명시해야 합니다. 이 외의 목적으로 복제, 배포, 출판, 전송 등 모든 유형의 사용행위를 하는 경우에 대하여는 사전에 허가를 얻고 비용을 지불해야 합니다.

작 설정 파일(default operation file)을 제공하고 모니터링 프로그램은 서비스가 시작될 때, 또는 서비스모듈 들이 구성될 때 각각의 서비스모듈 들이 기본 동작 설정 파일에 명시된 최소 요구 조건을 만족하는지 체크한다. 그리고 사용자에 최적화된 지능공간이 구성이 되고 작동할 때 정확하게 동작한 상황을 시퀀스 로그(Sequence log)를 남겨서 데이타베이스에 저장한다. 성공적인 구성의 유무는 모든 기본 동작(default operation)의 응답시간, 장치로부터 요구한 데이타 값의 정확성 등을 토대로 기본적인 서비스 구성의 성공 유무를 판별한다. 그리고 모니터링 시스템은 서비스 모듈 또는 지능공간이 동작 중 에러가 발생한 상황도 로그로 남겨둠으로서 유저가 만족한 서비스를 받은 성공적인 경우와 서비스 딜레이, 데이타 손실 등 실패의 경우를 가지고 비교, 판단함으로서 좀 더 지능적인 모니터링 시스템을 만든다. 또한 현재 서비스 구성에 부하가 많이 결렸을 경우 사용하지 않는 서비스(객체)를 정지시켜 자원의 여유를 둠으로서 자원의 점유율을 분산(Load balancing)하고 서비스 모듈의 재구성, 네트워크 대역폭 확보 등을 실행 한다.

본 논문에서는 이상과 같은 메시지 전송 속도와 이벤트 응답속도 등의 모니터링과 서비스 모듈의 기능적 동작 여부와 서비스의 상태 등을 수시로 모니터링하고 분석함으로서 시나리오나 서비스 상황에 따라 자원이나 서비스 모듈을 효과적으로 관리할 수 있는 유비쿼터스 모니터링 시스템을 제안한다.

2. 관련연구

본 논문의 모니터링 시스템은 로그의 분석한 결과를 가지고 정책결정을 함으로써 최적화를 수행한다. 다음의 정책 결정 관련 연구를 참고해서 최적화 방법을 모색했다.

2.1 ECA-P

유비쿼터스 공간에 모바일 장치가 동적으로 접속 되고 구성이 될 때 관리를 위해 정책(Policy)를 사용한다. 정책은 Event가 발생하고 정의된 조건이 만족 되었을 때 액션을 취하게 하는 정의된 규칙이다. 이것을 ECA (Event-Condition-Action)라고 정의한다. 그러나 정책의 충돌을 발견하고 해결하는 방법과 정책을 정확하게 수행하는 것을 보장하지 못한다. 그래서 위의 문제를 해결한 것이 ECA-P(Event-Condition-Action-Post-condition)이다[2]. ECA-P는 정책을 수행한 후 다음 조건으로서 시스템 상태를 지정한다. 즉, ECA규칙을 사용하면서 정책규칙을 분석하기 위해 Post-Condition을 명시한다.

ECA-P를 설명하는 그림 1은 다음과 같이 진행한다. 권한이 없는 Tom이 Active space[3]에 들어 왔을 때 규칙 R1은 권한 검사 프로그램이 작동하고 규칙 R2는

```
R1: On(ObjectEnter(Person P))
    if(Not_equal(role(p), "owner")}
    do(start(authorization_app))
    { status(authorization_app, running)}

R2: On(ObjectEnter("Tom"))
    if(true)
    do(stop_apps())
    { status(log_app, stopped),
      status(authorization_app, stopped)}
```

그림 1 ECA-P 정책

권한 검사 프로그램을 멈추게 한다. 그래서 충돌이 발생한다. 여기서 중괄호에 정의한 ECA-P는 액션이 발생하는 효과에 대해 기술해서 규칙간의 충돌을 감지할 수 있게 한다.

2.2 Four level-MCS

MCS(Monitoring and Control System)[4]은 중요한 도메인(domain)의 문제점이 발생했을 때 사용자에게 필요한 정책을 선택하도록 결정하는데 도와준다. 즉, 센서들에 의해 모니터된 정보를 모으고, 이 정보가 잘못된 것인지 평가한 후 잘못된 상황이면 적절한 액션을 취하는 것이다. Observation, Interpretation, Actuation으로 구성된 3개의 레벨(Level)로 되어있는 MCS는 센서의 오류나 외부 요소에 의한 잘못 해석된 데이타로 정확한 판단을 못할 수 있다.

그래서 그림 2에서 보듯이 Four Level-MCS는 Interpretation Level과 Actuation Level사이에 Correlation Level를 추가함으로서 이러한 문제를 해결했다. 이것은 서로 다른 센서에서 받은 해석된 정보를 모아 서로 연관 관계된 정보인지 필터링, 관리해서 Global View를 준다. 그리고 Global View를 통해 정확한 정책을 선택해서 컨트롤 액션을 취한다.

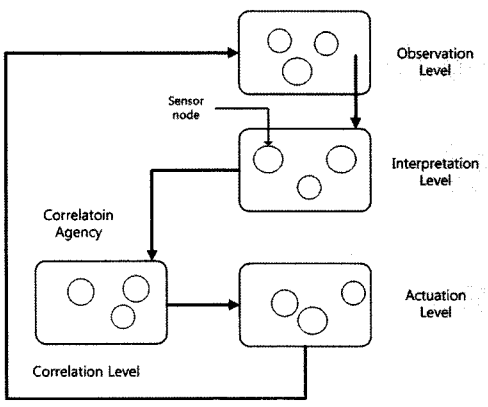

그림 2 Four Level-MCS 구조도

3. USPACC를 위한 모니터링 방법

3.1 모니터링 구조도

USPACC는 애플리케이션 프레임워크의 개발로 다양한 종류의 애플리케이션을 통합 관리하고 지능 공간에서 애플리케이션이 통일화된 방법으로 개발, 실행될 수 있는 환경을 제공한다. 그리고 사용자 중심의 서비스를 구현하기 위한 동적 커뮤니티 컴퓨팅 모델[1]은 고수준으로 사용자의 요구를 커뮤니티 서비스(Community Service)로 추상하여 사용자의 상황을 인지해서 자동으로 서비스 모듈들을 구성해 사용자에게 제공하는 시스템이다. 모니터링 대상별로 레벨을 나누어 모니터링 한다. 그림 3은 레벨별 모니터링 요소를 나타내고 있다.

각 레벨별로 모니터링하는 대상은 표 1과 같다.

USPACC를 위한 모니터링 시스템(그림 4)은 Level3의 커뮤니티서비스 및 Level2의 서비스모듈 등을 모니터링 하는 프레임워크(Monitoring Framework)를 제안하는 구조도이다. 기존에는 현재 모니터링한 상황, 그리고 에러정보 등을 등을 가지고 로그를 남겼다. 본 논문에서 제안하는 것은 단순히 로그만 남기는 것이 아니라 현재 모니터링하길 원하는 것의 문제점 및 성능 개선을 위한 목적으로 로그를 남긴다. 이 로그를 이용해 모니터링 대상의 에러원인을 찾고 성능 개선 기준점을 찾는다. 이것은 메소드(method)단위의 분석[5]으로 메소드가 불리는 시간, 모듈간의 데이타전달 상황 등을 로그로 남겨서 잘 동작한 경우와, 에러가 발생한 경우를 분석해서 사후에 같은 에러가 발생하지 않는 예측 가능한 시스템

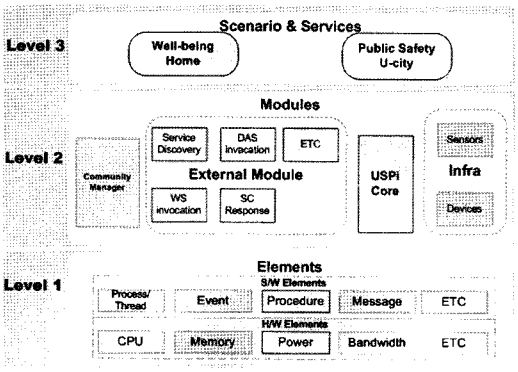

그림 3 유비쿼터스 시스템 모니터링 레벨

표 1 레벨별 모니터링 대상

Level1	컴퓨터 시스템의 자원을 수집하여 모니터링
Level2	서비스 상태를 메소드단위까지 Profiling해서 정보를 수집하여 모니터링
Level3	지능공간의 서비스 및 시나리오의 기능, 성능 상태를 모니터링

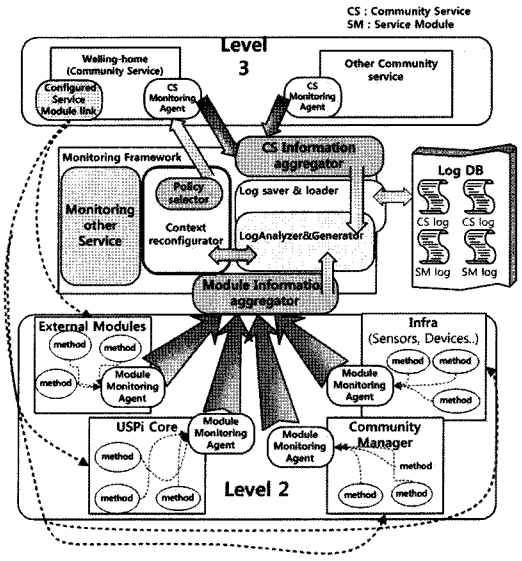

그림 4 모니터링 시스템 구조도

으로 만들 수 있다. 서비스 모듈이 처음 시작할 때 Module Information aggregator를 통해 기본 동작을 로그파일로 만든다. 그 다음 로그분석기와 로그생성기를 통해 서비스모듈을 위한 기술 로그(description log)를 생성하고 데이타베이스에 저장한다. 그리고 커뮤니티 서비스가 구성이 되면 처음에 각각의 서비스모듈을 현재 상태와 기본동작 로그파일을 가지고 동작 상태를 검사한다. 만약 상태 이상이 생기면 에러상황을 로그로 만들어 남긴다. 커뮤니티서비스가 처음 구성이 되었을 때 커뮤니티서비스에 속해 있는 모듈들의 실행 순서들을 만든다. 이것은 커뮤니티서비스 모니터링 에이전트의 실시간 모니터링을 통해 CS Information aggregator에게 모니터링 상황을 알려줌으로서 실행 순서들이 명시된 커뮤니티서비스 기술 로그로 데이타베이스에 저장된다. 그리고 각각의 서비스모듈이 성공적으로 동작하고 서비스를 제공하면 처음에 로그를 만들 때 등록하지 못한 서비스모듈들의 내부호출(nested call) 메소드를 기본 로그정보에 업데이트한다.

그리고 모니터링 시스템은 서비스가 동작되기 위한 최소한의 데이타 값인 속성 값을 보고 서비스모듈간의 기본 동작이 제대로 수행이 되는지 검사한다. 그리고 에러가 발생하면 로그로 남긴다. 속성 값은 개발자에 의해 작성 된다. 또한, 기존에 구성되어 졌던 커뮤니티서비스이면 기존 로그상황과 비교를 하 면서 성능이 좋아진 부분 또는 문제가 생기는 부분을 로그로 기록한다. 이렇게 만들어진 로그정보를 가지고 성능 및 자원 할당 상태를 파악해 Context reconfigurator는 현재 필요한 정책(Policy)을 선택해서 서비스모듈 성능개선 및 모듈 재

배치를 한다.

3.2 로그파일 저장형식

서비스모듈을 감시하는 모듈 에이전트(agent)와 커뮤니티서비스를 감시 하는 커뮤니티서비스 에이전트는 실시간으로 감시하며 정보가 업데이트 되면 로그 파일에 반영한다. 각 라인마다 수행시간을 표시하고 현재 모듈 또는 메소드 단위의 응답시간을 검사해서 성능상의 문제점 및 예측 가능한 시스템을 만들 수 있다. 로그저장 형식은 서비스모듈을 위한 저장 형식과 커뮤니티서비스를 위한 저장 형식으로 나뉘어서 데이타베이스에 저장한다. 이 정보를 바탕으로 Context reconfigurator는 상황에 맞는 정책을 정해서 Policy selector에 의해 성능을 개선한다.

3.2.1 서비스모듈 기술 로그 파일

모듈이 처음 모니터링에 등록이 되어서 시작되거나 또는 서비스 수행 중에 모듈 정보가 업데이트 된다면 그림 5와 같은 형식으로 저장이 되어서 사용된다. 로그 형식은 처음 시작에 모듈의 이름을 적는다. 그 다음 모듈의 시작을 알린다. 각 라인은 모듈 이름, 메소드인자 데이타, 리턴 값, 그리고 메소드가 호출된 시간을 기록한다. 만약 한 메소드에서 중첩된 메소드가 호출이 되면 중첩 메소드를 알리는 키워드를 넣어서 추가한다.

마지막을 알리는 키워드로 로그 끝을 알리고 이 모듈이 실행된 전체 시간을 기록한다. 다음 그림 6은 서비스모듈 기술 로그 파일의 형식을 설명한다.

아래 표 4는 서비스모듈 로그 기술을 설명한다.

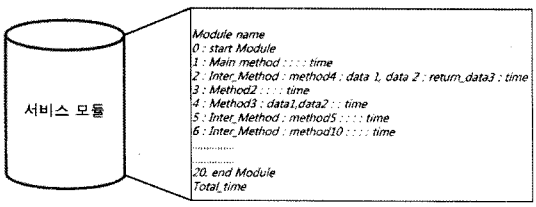

그림 5 서비스모듈 기술 로그 파일 형식 예제

```
call_number : method_name : parameter_list : return_data
                                              : time_stamp
keyword => Inter_ , Module_name, start_Module,
           end_Module , Total_time, error_
```

그림 6 서비스모듈 기술 로그 파일 형식

표 4 서비스모듈 로그 기술

Call_sequence	method가 불리는 순서
method_name	method 이름
parameter_list	method에 넘어가는 parameter값
return_data	method에서 나오는 결과 값
time_stamp	method가 실행 됐을 때를 기록한다.

표 5 서비스모듈 키워드 설명

*Inter_*	method이름 앞에 *Inter*라고 붙이면 바 로 전의 method안에서 또 다른 method가 call된 것을 의미한다.
*error_*	모듈 이름 앞에 에러를 붙임으로서 현 재 에러가 난 지점을 알 수 있다.
Module_name	묘듈의 이름
start_module	log 시작
end_module	log 끝
Total_time	모듈이 수행된 시간

서비스모듈 로그 기술의 키워드는 표 5와 같다.

3.2.2 커뮤니티서비스 기술 로그 파일

서비스 모듈들이 커뮤니티 서비스를 구성하고 사용자에게 서비스를 제공할 때 서비스 모듈 간에 데이타를 주고받는다.

그때 모듈에서 보낸 데이타를 모듈 호출 순서대로 로그에 기록 한다. 형식은 처음에 현재 서비스 중인 커뮤니티서비스이름을 나타내고 서비스 시작을 기록한다. 그리고 현재 수행중인 모듈은 다른 모듈로 부터 데이타를 받거나 현재 모듈에서 데이타를 보낼 때 데이타 값과 데이타를 받은 시간을 기록한다. 그리고 모듈이 끝났을 때 현재 수행중인 모듈이 수행된 전체 시간을 기록한다. 모듈의 호출이 끝났을 때 서비스 종료를 알리고 커뮤니티서비스전체 수행된 시간을 기록한다. 그림 7에서 기록된 로그는 그림 8과 같은 형식으로 저장이 된다.

다음 표 6은 커뮤니티서비스 로그 기술을 설명한다.

커뮤니티서비스 로그 기술의 키워드는 표 7과 같다.

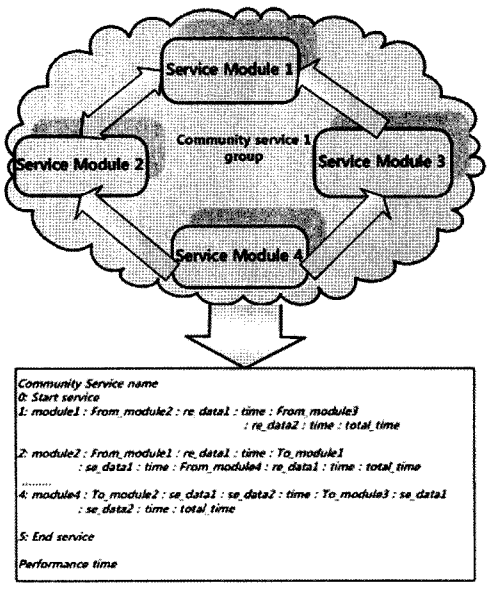

그림 7 커뮤니티서비스 기술 로그 파일 형식 예제

표 6 커뮤니티서비스 로그 기술

Call_number	모듈이 호출되는 순서	
Module_name	현재 동작하는 모듈	
S	서로 연관된 또 다른 서비스 모듈, 복수개의 모듈에게 데이타를 전달하거나 전달 받는다.	
	module_name	모듈 이름 앞에 키워드를 붙여서 data가 어디서 오는지 그리고 어디에 데이타를 보내는지 식별
	D	복수개의 데이타에 키워드를 붙여서 데이타 를 주고받는 걸 알 수 있다.

```
D = data
S = module_name : D1 : D2 : ... : Dn : data delivery time
call_number : current_module : S1 : S2 : ... : Sn
                                : total delivery time
keyword => From_ , To_ , se_ , re_ error_
```

그림 8 커뮤니티서비스 기술 로그 파일형식

표 7 커뮤니티서비스 키워드

*From_*	data를 받은 모듈
*To_*	data를 보낼 모듈
*se_*	보내는 data
*re_*	받은 data
*error_*	현재 에러가 난 모듈을 기록

4. 성능 개선을 위한 정책 규칙

로그 분석기는 현재 로그정보를 분석해서 서비스모듈 또는 커뮤니티서비스의 성능의 문제점을 찾는다. 그리고 Context reconfigurator가 로그 분석한 결과를 가지고 서비스 모듈의 재배치 및 필요 없는 서비스 중단, 네트워크 상황을 검사하고 Policy selector에 의해 정책을 결정해서 성능을 개선한다. 이 정책 규칙은 개발자에 의해 작성되는 것이다. 그림 9는 서비스모듈 또는 커뮤니티서비스의 정책 규칙 예이다. R1의 규칙은 서비스모듈이 수행 중에 네트워크상황이 좋지 않기 때문에 현재 네트워크를 점유 중인 모듈에게 사용을 잠시 중지하라고 시킨다.

R2의 규칙은 현재 구성중인 커뮤니티서비스가 성능이 좋지 않으니 서비스모듈의 연결을 재배치함으로써 응답시간을 높인다.

```
R1 : if( state( SM, Net_Speed_Low ) )
     action( Net, Band_TuneUP )
R2 : if( state( CS, Performance_Low ) )
     action( SM_Config, Re_Arrange ) )
```

그림 9 성능개선을 위한 정책 규칙

5. 결 론

최근 유비쿼터스 시스템에 관한 많은 기술이 나오고 있다. 그러나 유비쿼터스 시스템이 일상생활에 가까워짐에 따라 응답성과 신뢰성이 기반이 되어야함에도 불구하고 기존 연구는 새로운 패러다임 제안과 기능제공을 위주로 수행되었다. 그래서 본 논문에서는 유비쿼터스 시스템의 활용성을 높이기 위하여 응답성, 신뢰성, 그리고 자원 효율성을 위해 로그정보를 저장하고 이를 이용해서 정책을 결정하는 모니터링 시스템을 제안했다. 현재 로그를 구성을 하고 있으나 향후 성능 저하가 없게 로그를 기록, 재 취합하고 비교하는 시스템을 개발 할 것이다. 그리고 정책 규칙은 개발자에 의해 작성이 되도록만 되어 있으나 규칙의 사용빈도와 로그의 기록들을 통해 자동으로 새로운 정책 규칙을 만드는 시스템을 개발 할 것이다.

참 고 문 헌

[1] Y.N. Lee, J.T. Lee and M.K. Kim "Multi-agent based Community Computing System Development with the Model Driven Architecture," Autonomous Agents and Multi-Agent Systems, May 2006.

[2] .shiva shankar, A.Ranganathan, and R.Campbell, "An ECA-P Policy-based Framework for Managing Ubiquitous Computing Environments," MobiQuitous, July 2005.

[3] A.Rangnathan, et al., "Mobile Polymorphic Applications in Ubiquitous Computing Environment," Mobiquitous, August 2004.

[4] S.Bandini, A.Mosca and M.Palmonari, "A Conceptual Framework for Monitoring and Contorl System Development," Ubiquitous Mobile Information and Collaboration Systems, June 2004.

[5] 류동항, 정민수, "자바 바이트 코드 분석기의 설계 및 구현", 한국정보과학회 '98 봄 학술발표논문집(A), 제25권 제1호, pp.77~79, 1998.