유비쿼터스 컴퓨팅 환경을 위한 MDA 기반의 어플리케이션 개발을 지원하는 통합개발환경

IDE based on MDA for Application Development in Ubiquitous Computing Environment

김동욱, Dongwook Kim*, 김준영, Junyoung Kim*, 이정태, Jungtae Lee**, 김민구, Minkoo Kim**
*아주대학교 정보통신전문대학원 정보통신공학과,
**아주대학교 정보통신대학 정보 및 컴퓨터공학부

요약 유비쿼터스 컴퓨팅 환경에서 커뮤니티 메타포를 이용하여 유비쿼터스 서비스를 실현하는 커뮤니티 컴퓨팅에 대한 연구가 제안되었다. 제시된 커뮤니티 컴퓨팅 어플리케이션 개발과정은 MDA 개발방법을 적용하여 추상화된 고수준의 모델로부터 최종 플랫폼에 적합한 코드를 생산한다. 그러나 커뮤니티 컴퓨팅 시스템을 구현하기 위한 통합개발환경은 존재하지 않는다. 이에 본 연구에서는 최근 관심이 고조되고 있는 이클립스를 이용하여 커뮤니티 컴퓨팅에서 요구되는 기능들을 플러그인으로 통합하여 커뮤니티 컴퓨팅 어플리케이션 개발에 적합한 통합개발환경(CDTK : Community computing Development Tool Kit)을 제안한다.

핵심어: Community Computing, Ubiquitous Computing, Eclipse, Plug-In, IDE, Tool

1. 서론

커뮤니티(Community) 메타포를 이용하여 유비쿼터스 서비스를 실현하고자 하는 커뮤니티 컴퓨팅이 제안되었다. 커뮤니티 컴퓨팅이란 유비쿼터스 지능 공간(USS:Ubiquitous Smart Space)을 정의하고, 공간내에 존재하는 각종 엔티티(Entity)들 간의 상호협력을 통하여 유비쿼터스 서비스를 구현하는 것이다.

제시된 커뮤니티 컴퓨팅 시스템의 개발과정은 MDA(Model Driven Architecture) 개발방법을 적용하여 추상화된 고수준의 모델로부터 모델변환과정을 거쳐서 최종 플랫폼에 적합한 코드를 생산하고 있다.[1,2] 그러나 이러한 커뮤니티 컴퓨팅 시스템의 구축에 효율적인 통합개발환경은 존재하지 않기 때문에 기존의 벤더에서 제공하는 통합개발환경에 최소한의 기능을 부가하여 어플리케이션 개발에 이용하였다.

최근 관심이 고조되고 있는 이클립스는 오픈소스 프로젝트, 공개표준 런타임, 다양한 기능을 지원하는 플러그인을 제공하는 등의 장점을 가지고 있기 때문에 통합개발환경을 구성하기 위한 적절한 방안으로 간주된다.[3,4]

본 연구에서는 커뮤니티 컴퓨팅 시스템의 구축에 공통적으로 요구되는 기능을 분석하여 이클립스를 기반으로 하는 통합개발환경(CDTK:Community computing Development Tool Kit)의 형태를 제안한다. 본 논문의 구성은 다음과 같다. 2장에서 커뮤니티 컴퓨팅 어플리케이션의 개발과정을 분석하고, 3장에서 이클립스를 기반으로 하는 통합개발환경을 제안한다. 마지막으로 4장에서 결론 및 향후 연구방향에 대하여 제시한다.

2. 커뮤니티 컴퓨팅

커뮤니티 모델링 언어(CML:Community computing Modeling Language)를 사용하여 커뮤니티 컴퓨팅 모델(CCM:Community Computing Model)을 기술한다. CCM에는 USS의 정의와 커뮤니티의 타입인 목표(Goal), 역할(Role), 상호협력을 위한 규약등이 기술된다. CIM-PI(Platform Independent Community Computing Implementation Model)는 커뮤니티의 목표,를 달성하기 위

본 연구는 21세기 프론티어 연구개발사업의 일환으로 추진되고 있는 정보통신부의 유비쿼터스컴퓨팅및네트워크원천기반기술개발사업의 지원에 의한 것임

한 엔티티들간의 통신방법을 추가로 기술한다. CIM-PS(Platform Specific Community Computing Implementation Model)는 CIM-PI를 특정 플랫폼에서 동작할 수 있도록 모델을 코드로 변환한다. 그림 1은 전체 개발과정이다.[1,5]

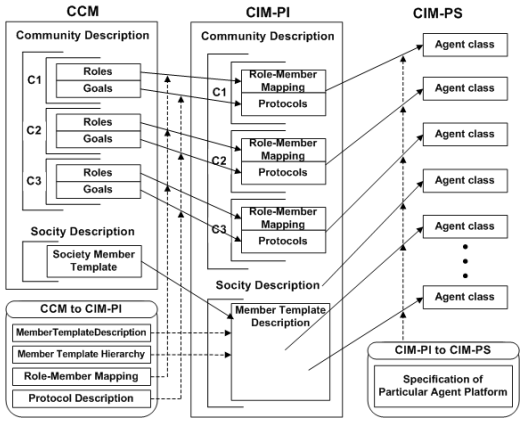

그림 1. 커뮤니티 컴퓨팅 개발과정

3. 통합개발환경

3.1 목적

기존 커뮤니티 컴퓨팅 개발과정은 다음과 같은 한계를 가지고 있었다.

첫째, 기존 통합개발환경을 사용하여 모델을 기술하고자 하는 경우에 모델을 텍스트 기반으로 기술하였다. 이는 유비쿼터스 지능 공간내에 존재하는 커뮤니티와 멤버, 역할과 그들간의 관계를 파악하는 것이 용이하지 않았다. 또한 새로운 모델을 기술하고자 하는 경우에 반복 코드를 작성해야 함으로써 모델의 재사용성을 저하시켰고, 모델이 제대로 기술되었는지 검증할 수 없었다.

둘째, 일관성 있는 작업 수행이 어려웠다. 커뮤니티 컴퓨팅의 개발과정인 CCM, CIM_PI, CIM_PS의 각 단계에서 수행해야 하는 기능이 다름에도 불구하고, 퍼스펙티브를 분리하지 않았기 때문에 커뮤니티 컴퓨팅의 개발과정에 요구되는 모든 기능들이 하나의 워크벤치에서 개발자에게 보여지게 되었다.

셋째, 플랫폼 종속적인 코드 생산 시에 유연성이 미비하였다. 기존의 통합개발환경은 JADE 플랫폼에 기반한 결과물을 생산함으로써 JADE 이외의 다양한 플랫폼에서 동작 가능한 코드를 생산하고자 하는 경우에 새로운 개발과정이 필요하게 되었다.

넷째, 기존 개발과정에서 최종적으로 생산된 결과물은 유비쿼터스 지능 공간에 존재하는 각 엔티티의 환경정보가 고려되지 않았기 때문에 최종 결과물을 실제 엔티티에 이식하여 동작하기를 기대하기 어려웠다.

위와 같은 문제점을 해결하기 위하여 개선된 통합개발환경이 필요하게 되었다.

3.2 기능 및 범위

개선된 통합개발환경에서 제공해야 하는 기능과 활용 범위는 다음과 같다.

첫째, 텍스트 또는 그래피컬하게 모델링이 가능한 커뮤니티 컴퓨팅 모델링 에디터를 제공하여 커뮤니티 컴퓨팅의 모델을 기술할 수 있게 한다. 그래피컬하게 모델을 기술함으로써 커뮤니티, 멤버 관련 정보들의 관계를 쉽게 파악할 수 있고, 모델의 재사용성이 높아질 것이다. 또한 커뮤니티 모델링 언어에 적합하지 않은 코드를 사전에 차단할 수 있게 됨으로써 잘 정의된 모델의 생성이 가능하게 된다.

둘째, 커뮤니티 컴퓨팅을 구현하기 위한 각 단계별 내부 변환 기능을 제공한다. 초기모델을 플랫폼 독립적인 모델로의 변환, 플랫폼 종속적인 모델로의 변환, 최종 코드로 변환하는 기능을 제공한다. 이러한 모든 기능은 각각 플러그인으로 개발되어 통합개발환경에 통합된다.

셋째, 각 단계별 변환단계에 필요한 요구사항을 파악하여 퍼스펙티브를 분리한다. 이를 통하여 각 단계에 필요한 기능들로 구성된 사용자 인터페이스를 커스터마이즈하여 제공하여 개발자에게 최적의 환경을 제공한다. 퍼스펙티브를 분리하였기 때문에 해당 단계에 필요한 기능만을 혼돈 없이 사용할 수 있게 된다.

넷째, 기존 통합개발환경과 달리 개선된 통합개발환경은 이클립스 플랫폼을 기반으로 함으로써 이클립스에서 제공하는 모든 기능을 재사용 할 수 있고, 오픈 소스로 제공되는 다양한 플러그인들도 필요에 따라 추가로 사용할 수 있게 된다.

다섯째, 최종 결과물은 이클립스 리치 클라이언트 플랫폼을 기반으로 한 엔티티에 이식되어 실제 동작이 가능하게 된다.

여섯째, 모델의 확장 및 멤버 어플리케이션의 동작에 필요한 모듈의 변경시에 이를 플러그인으로 개발하여 확장함으로써 새로운 통합개발환경으로 업그레이드가 가능하다.

3.3 최종 결과물

개선된 통합개발환경에서 도출하고자 하는 결과물은 소사이어티 매니저(Society Manager), 커뮤니티 매니저

(Community Manager), 그리고 멤버 어플리케이션(Member Application)이다.

3.3.1 소사이어티 매니저

소사이어티 매니저는 커뮤니티 매니저를 생성하고, 실행중인 커뮤니티 매니저 정보를 관리한다. 또한 커뮤니티 컴퓨팅에 참여하는 멤버의 등록 및 탈퇴를 관리한다.

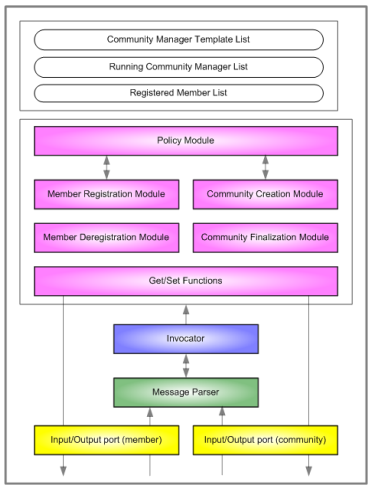

그림 2. 소사이어티 매니저

3.3.2 커뮤니티 매니저

커뮤니티 매니저는 커뮤니티 내의 각 컴퓨팅 자원들의 정보를 관리한다. 롤의 정보를 관리하고, 롤과 멤버의 결합조건을 만족하는 멤버 캐스팅을 수행하여 선택된 멤버들의 정보를 유지한다. 또한 해당 멤버에게 실제로 작업수행을 지시한다.

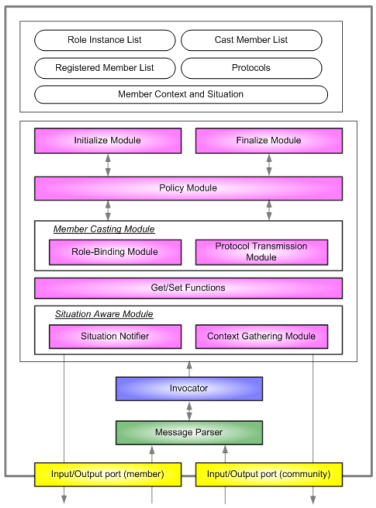

그림 3. 커뮤니티 매니저

3.3.3 멤버 어플리케이션

이기종 환경하에 존재하는 실제 엔티티에 이식 가능한 프로그램으로써 커뮤니티의 멤버로 참여하여 멤버간 협업을 통한 유비쿼터스 서비스를 실현하게 한다.

3.4 전체 구조

통합개발환경은 최종 결과물인 소사이어티 매니저, 커뮤니티 매니저, 멤버 어플리케이션을 생성하기 위하여 다음과 같은 전체 구조를 가지게 된다. 전체 구조에서 각 개발단계는 스페이스 모델링 단계(Space Modeling), 플랫폼 독립적인 모델 단계(Platform Independent Implementation), 플랫폼 종속적인 모델 단계(Platform Specific Implementation), 실행단계(Run)로 구분된다.

3.4.1 스페이스 모델링 단계

커뮤니티 컴퓨팅 모델을 모델링 에디터를 이용하여 모델을 기술하는 단계이다. 스페이스 모델링 단계의 목적은 유비쿼터스 지능 공간을 정의하고 공간내에 존재하는 엔티티들을 정의함으로써 커뮤니티 컴퓨팅 모델을 기술하는 것이다.

3.4.2 플랫폼 독립적인 모델 단계

기술된 커뮤니티 컴퓨팅 모델을 구현수준의 플랫폼 독립적인 모델로 확장하는 단계이다. 이전 단계에서 기술되지 않은 멤버들 간의 협업과정에 대한 기술을 구체화한다.

3.4.3 플랫폼 종속적인 모델 단계

플랫폼 종속적인 최종 코드를 생산하기 위한 단계이다. 플랫폼 독립적인 모델로부터 소사이어티 매니저와 커뮤니티 매니저를 도출하고, 이식할 엔티티의 환경정보를 입력받아서 실제로 엔티티에 이식할 수 있는 멤버 어플리케이션을 생성한다.

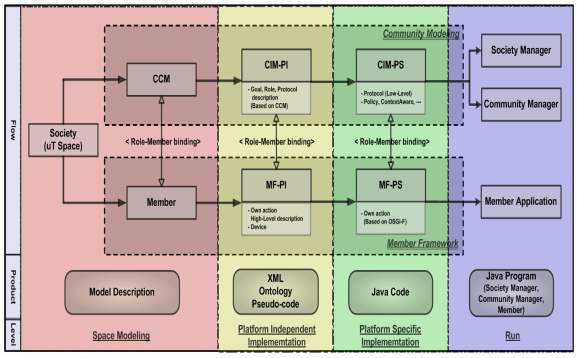

그림 4 통합개발환경의 전체 구조

3.5 멤버 프레임웍

3.5.1 멤버 어플리케이션

유비쿼터스 지능 공간내에 존재하는 엔티티들은 본래 자신의 고유한 역할을 수행한다. 이후에 커뮤니티의 목표를 달성하기 위하여 커뮤니티의 멤버로 캐스팅 되는데 이때 멤버화를 위하여 이식되는 어플리케이션을 멤버 어플리케이션이라 한다.

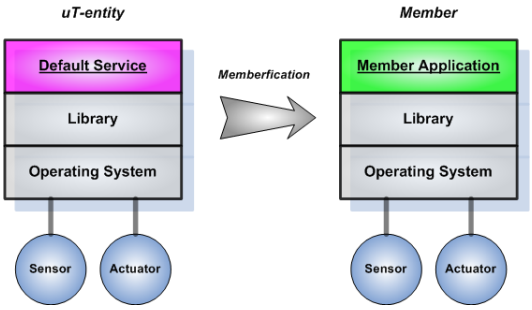

그림 5. 엔티티의 멤버화 과정

3.5.2 소사이어티 멤버와 커뮤니티 멤버

엔티티의 멤버화 과정은 소사이어티 멤버화(Society Memberfication)와 커뮤니티 멤버화(Community Memberfication)로 구분된다. 유비쿼터스 지능 공간내에 존재하는 다양한 엔티티들은 특정 소사이어티에 포함되면 소사이어티 멤버화가 된다. 이때 멤버화된 엔티티에는 멤버 프레임웍의 고정된 부분이 이식되고, 이후에 커뮤니티의 멤버로 캐스팅되면 추가로 다운로드 되어 멤버로써의 기능을 수행하게 된다.

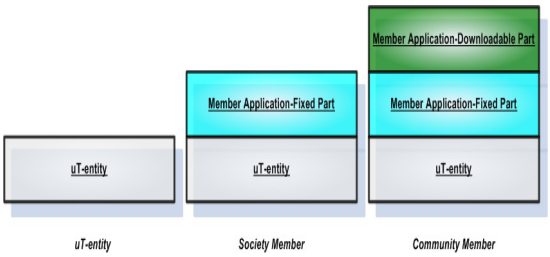

그림 6. 소사이어티 멤버와 커뮤니티 멤버

3.5.3 멤버 프레임웍

통합개발환경을 사용하여 생성한 어플리케이션을 실제 엔티티에 이식하여 동작하기 위해서는 각 엔티티의 종류에 상관없이 커뮤니티 컴퓨팅 어플리케이션이 이식될 수 있는 환경이 필요하다. 엔티티가 커뮤니티의 멤버로 참여하여 멤버간의 협업을 통하여 유비쿼터스 서비스를 실현할 수 있도록 지원하는 어플리케이션 프레임웍을 멤버 프레임웍이라고 한다.

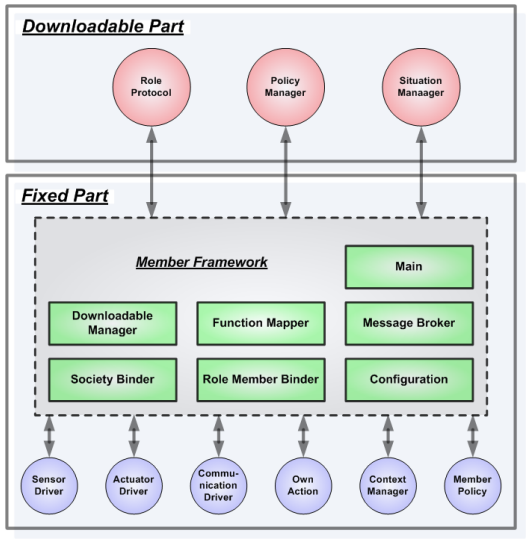

그림 7. 멤버 프레임웍 아키텍처

멤버 프레임웍의 내부 컴포넌트의 대략적인 기능은 다음과 같다.

- Main : 플러그인을 초기화하여 멤버 어플리케이션을 구동하는 기능을 한다.
- Configuration Manager : 멤버가 실행되기 위하여 필요한 자원과 구성요소를 관리하는 기능을 한다.
- Downloadable Manager : 커뮤니티 매니저로부터 넘겨받은 정보를 관리하는 기능을 한다.
- Role-Member Binder : 소사이어티 멤버가 멤버 캐스팅과 롤 프로토콜 다운로드를 통하여 커뮤니티 멤버가 되기 위한 연결 기능을 한다.
- Society Binder : 소사이어티 매니저에게 멤버의 등록을 요청하고, 소사이어티 매니저로부터 멤버의 아이디를 수신하여 멤버화를 수행한다. 또한 커뮤니티 매니저에게 커뮤니티의 생성을 요구하는 기능을 한다.
- Message Broker : 메시지를 송수신하기 위한 버퍼 기능을 한다.
- Function Mapper : 롤 프로토콜과 자신이 행할 수 있는 서비스를 맵핑하고, 롤 프로토콜의 서비스 수신 관리를 한다. 서비스와 디바이스에 대한 기술정보와 탐색하는 기능을 한다.

엔티티에 미리 이식되어 있어야 하는 고정부분(Fixed Part)의 기능은 다음과 같다.

• Sensor Driver : 엔티티에 장착된 센서 디바이스를 작동하는 드라이버 관리 기능을 수행한다. 센서로부터 읽어 들인 로우 데이터를 컨텍스트 매니저에게 전달한다.

• Actuator Driver : 엔티티에 장착된 액츄에이터 디바이스를 작동하는 드라이버 관리 기능을 수행한다. Own Action 플러그인으로 부터 받은 액션을 수행한다.

• Communication Driver : 커뮤니티 매니저, 커뮤니티 멤버등과의 통신기능을 수행하고, 롤 프로토콜이 보내는 메시지를 받아서 다른 커뮤니티의 커뮤니케이션 드라이버에게 전달한다. 또한 다른 커뮤니케이션 드라이버로부터 받은 메시지를 컨텍스트 매니저에게 전달한다.

• Context Manager : 센서로부터 받은 로우 데이터의 필터 기능을 하여 컨텍스트를 생성한다. 커뮤니케이션 드라이버로부터 들어온 외부 정보를 관리한다.

• Member Policy ; 커뮤니티 매니저의 멤버 캐스팅 요청에 대하여 멤버가 이를 받아 들일지를 결정하는 기능을 수행한다.

• Own Action : Own Action의 리스트를 관리하고 실행 상태를 관리한다. 롤 프로토콜의 서비스 수행을 위하여 서비스 디스크립션을 확인하여 액션을 순서대로 진행한다.

커뮤니티의 멤버로 캐스팅 되었을 때 추가로 다운받아야 하는 부분(Downloadable Part)의 기능은 다음과 같다.

• Role Protocol : 커뮤니티 매니저로부터 생성된 프로토콜을 해석하여 실행한다.

• Policy Manager : 커뮤니티 매니저로부터 받은 정책정보를 이용하여 컨텍스트 필터링을 수행한다.

• Situation Manager : 고정부분의 컨텍스트 매니저에서 생성된 컨텍스트를 전달받아서 상위의 소사이어티에게 전달하는 기능을 한다.

3.6 통합개발환경의 구성방안

3.6.1 각 단계별 퍼스펙티브의 분리

커뮤니티 컴퓨팅 모델을 생성하기 위한 기능은 스페이스 모델링 단계에 필요하다. 스페이스 모델링 단계를 위한 퍼스펙티브를 선택하면 활성화되어 있는 모델링 메뉴를 사용할 수 있게 한다.

이클립스 기반의 통합개발환경에서 모델링 에디터가 탑재된 모델링 단계의 퍼스펙티브를 생성하기 위하여 플러그인 설명서(plugin.xml)에 퍼스펙티브의 확장점을 선언하고 해당 인터페이스를 구현하는 클래스를 생성한다. 인터페이스가 제공하는 메서드를 사용하여 뷰나 편집기의 초기 배치를 기술하는 것이 가능하여 아래 그림과 같이 사용자 인터페이스를 커스터 마이즈 할 수 있게 된다.

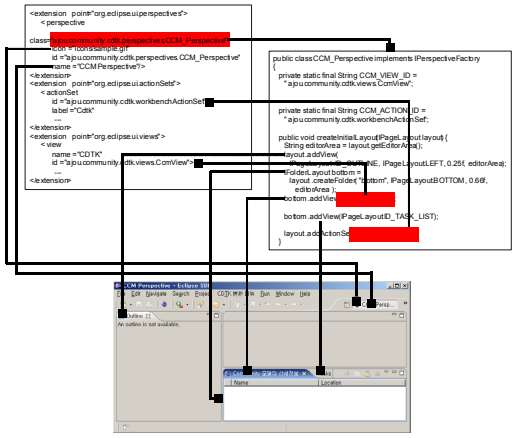

그림 8. 스페이스 모델링 단계의 퍼스펙티브

3.6.2 모델링 에디터 제공

일반적인 파일 탐색 트리와 유사한 모델링 에디터를 사용하여 커뮤니티 관련정보인 소사이어티, 커뮤니티, 롤, 골, 멤버를 트리 형태로 보여주게 된다. 아래 그림과 같은 모델링 관련 컨텍스트 메뉴를 클릭하였을 경우에 보여지는 템플릿 형태의 에디터를 입력하는 것으로 트리를 생성할 수 있게 한다.

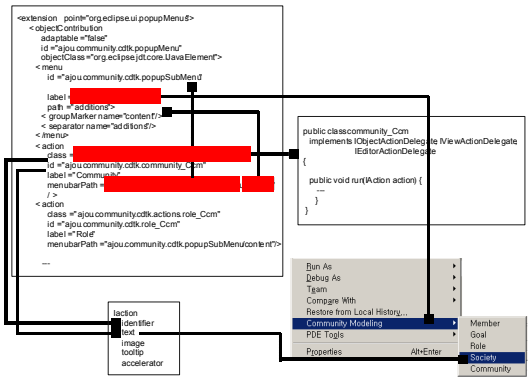

그림 9. 커뮤니티 컴퓨팅 모델링 메뉴구성

커뮤니티 관련 정보의 입력이 완료된 후에 커뮤니티 컴퓨팅 모델로의 변환 명령을 수행하면 다음과 같이 스크립트 형태의 모델이 생성되고, 플랫폼 독립적인 모델 단계의 퍼스펙티브로 이동하게 된다.

```
Society {
Community Community_Type_Name {
  Role Role_Name
    Attribute:
     {attribute name={attribute values | data type}}+
    Action: Action_Name {action description};
    Cast: {attribute name=attribute values}+
  Goal {Goal_Name (participant role names)
    {
      situation1:  'what to do'  description ;
      situation2:  'what to do'  description ; …} }+ }
Member Type Society Member {
  Attribute:
    {attribute name={attribute values | data type}}+
    Action: Action_Name {action description};
  Society Policy{
    Community Precedence=
     {1st={community_name,..}, 2nd={..},….}
    Exclusive Community =
     {(Community_Type_Name,Community_Type_Name )*}
    Member Casting:
     { Member Casting Policy Description}
    Member Management:
     { Member Management Description}}
}
```

위와 같은 방법으로 각 단계별 필요한 기능들을 플러그인으로 개발하고, 각 단계별 퍼스펙티브를 분리하면 된다.

3.6.3 구현시 요구되는 플러그인들의 결합

본 연구에서 제안하는 통합개발환경에 요구되는 각 플러그인들을 결합하여 이클립스 플랫폼에 통합하면 다음과 같은 구조가 된다.

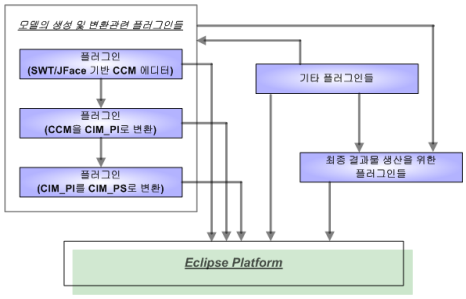

그림 10. 요구되는 플러그인들의 통합

4. 결론 및 향후 연구방향

본 연구는 커뮤니티 컴퓨팅 시스템을 구현하기 위한 개선된 통합개발환경을 제안하였다. 개선된 통합개발환경을 이용하면 커뮤니티 컴퓨팅 시스템 구현에 필요한 모델의 생성, 모델로부터 코드로의 변환, 엔티티간의 동작을 위한 어플리케이션의 생성과 같은 기능을 제공 받게 된다. 이를 통하여 개발 시간의 단축, 개발업무의 분업화, 개발과정의 연계성을 확보할 수 있게 되고 실제 엔티티에 이식가능한 코드를 생산할 수 있을 것으로 판단된다. 또한 통합개발환경은 이클립스를 기반으로 함으로써 향후 예상되는 추가적인 모델의 확장, 변경 및 멤버 프레임웍의 모듈 수정등이 발생할 경우에도 유지 보수가 용이하다. 변경된 부분만을 플러그인으로 추가적으로 개발하여 기존 플러그인을 교체하면 추가적인 요구사항을 충족하는 개선된 통합개발환경이 될 것이기 때문이다.

현재 본 논문에서 제안한 통합개발환경의 구현이 진행중이며, 향후 이를 활용하여 실제로 로봇등에 이식하여 검증하는 작업이 필요하다.

참고문헌

[1] Yuna Jung, Jungtae Lee, Minkoo Kim. "Multi-agent based Community Computing System Development with the Model Driven Architecture", AAMAS, 2006.

[2] OMG, Model Driven Architecture, "MDA Guide Version 1.0.1", http://www.omg.org/mda, June 2003.

[3] Eclipse.org, "Eclipse Platform Technical Overview", http://www.eclipse.org/articles/Whitepaper

[4] Jim D'Anjou, Scott Fairbrother et al, "The Java Developer's Guide to Eclipse-Second Edition", 2004.

[5] 김동욱, 이정태, 류기열, "커뮤니티 컴퓨팅 어플리케이션 개발을 지원하기 위한 이클립스 기반의 통합개발환경", 한국컴퓨터종합학술대회, 2005.