논리적 셀 기반의 로봇 소프트웨어 컴포넌트 저장소

(A Logical Cell-Based Approach for Robot Component Repositories)

구 형 민 † 고 인 영 ††
(Hyung-Min Koo) (In-Young Ko)

요 약 다양한 환경에 배치될 수 있고, 예상치 못한 상황에 자주 접할 수 있는 지능형 서비스 로봇의 경우에는 처할 수 있는 환경과 상황을 모두 예측하여 로봇 내부에 필요한 기능을 모두 가지고 있기 어렵다. 로봇에게 환경에 맞는 필요한 기능만 내부에 가지고 있을 수 있게 하고, 필요에 따라 새로운 기능을 획득할 수 있도록 지원해 주기 위한 기반 기술인 컴포넌트 저장소가 본 논문의 주제이다. 이 저장소를 실제 로봇 플랫폼에 적용 실험 해 온 결과, 로봇이 일일이 외부 저장소들을 접근함에 따라 필요한 컴포넌트의 검색 및 획득의 성능이 저하된다는 문제와, 가용한 컴포넌트 저장소가 늘어감에 따라 확장성, 공유성 문제가 발견되었다. 본 연구에서는 이러한 문제점들을 해결하기 위해 분산된 컴포넌트 저장소들을 컴포넌트의 기능적인 측면에 따라 논리적인 그룹으로 묶은 셀 기반의 진화적인 컴포넌트 저장소를 개발하였다. 프로토타입을 개발하여 실험한 결과, 셀 기반의 저장소를 이용하여 로봇이 분산된 저장소를 일일이 물리적으로 접근하는 것이 아니라 논리적으로 투명하게 접근을 할 수 있도록 지원한다. 또한, 로봇 컴포넌트/애플리케이션 개발자들이 자신의 접근 가능한 저장소를 변경하면 전체 저장소 시스템에 반영되어 다른 개발자들이나 로봇이 사용할 수 있게 해 주는 컴포넌트의 투명한 공유를 지원 한다.

키워드 : 지능형 서비스 로봇, 자가성장 로봇 소프트웨어, 분산 온톨로지 저장소, 분산 컴포넌트 저장소

Abstract Self-growing software is a software system that has the capability of evolving its functionalities and configurations by itself based on dynamically monitored situations. Self-growing software is especially necessary for intelligent service robots, which must have the capability to monitor their surrounding environments and provide appropriate behaviors for human users. However, it is hard to anticipate all situations that robots face with, and it is hard to make robots have all functionalities for various environments. In addition, robots have limited internal capacity. To support self-growing software for intelligent service robots, we are developing a cell-based distributed repository system that allows robots and developers transparently to share robot functionalities. To accomplish the creation of evolutionary repositories, we invented the concept of a cell, which is a logical group of distributed repositories based upon the functionalities of components. In addition, a cell can be used as a unit for the evolutionary growth of the components within the repositories. In this paper, we describe the requirements and architecture of the cell-based repository system for self-growing software. We also present a prototype implementation and experiment of the repository system. Through the cell-based repositories, we achieve improved performance of self-growing actions for robots and efficient sharing of components among robots and developers.

Key words : Intelligent Service Robots, Self-Growing Robot Software, Distributed Ontology Repositories, Distributed Component Repositories

· 이 연구(논문)는 산업자원부 지원으로 수행하는 21세기 프론티어 연구개발사업(인간기능 생활지원 지능로봇 기술개발사업)의 일환으로 수행되었습니다.
† 학생회원 : 한국정보통신대학교 공학부
hyungminkoo@icu.ac.kr
†† 정 회 원 : 한국정보통신대학교 공학부 교수
iko@icu.ac.kr
논문접수 : 2007년 4월 25일
심사완료 : 2007년 6월 28일

1. 서 론

지능형 서비스 로봇(Intelligent Service Robot)이란 스스로 환경을 모니터링하고, 문제 상황 발생시 이를 극복하기 위한 방안을 마련하여 적절한 행위를 할 수 있는 로봇을 말한다[1]. 지능형 서비스 로봇은 예기치 못한 환경이나 문제에 자주 직면할 수 있는데 모든 상황을 미리 예견하여 그에 필요한 기능을 로봇이 모두 갖추도록 하는 것은 불가능 할 것이다. 따라서 로봇에게는 환경을 모니터링 하여 필요한 기능과 그 기능의 조합을 필요에 따라 외부에서 획득하여 기능을 성장해 나가는 자가 성장 소프트웨어가 필요하다[2]. 본 연구는 자가 성장 소프트웨어를 지원해 주기 위한 기반 기술인 소프트웨어 컴포넌트 저장소 시스템을 개발하기 위한 것이다.

컴포넌트 저장소는 그림 1과 같이 로봇에게는 외부 저장소들을 이용하여 다양한 기능의 컴포넌트를 획득할 수 있도록 지원해 준다. 즉, 로봇 내부에는 환경에 적합한 꼭 필요한 기능들을 유지하도록 지원하고 필요에 따라 외부로부터 필요한 기능을 탐색, 수집할 수 있도록 지원하며 로봇간 컴포넌트의 공유를 지원한다.

로봇 컴포넌트 및 애플리케이션 개발자들에게는 컴포넌트 등록, 검색을 지원하여 컴포넌트를 공유하게 함으로써 컴포넌트의 재사용성을 증가시켜 주고, 분산되어 있는 개발자들이 개발한 컴포넌트들을 로봇이 실행시간 동안 사용할 수 있도록 지원해 준다.

로봇의 실시간 기능 획득이 가능한지 적용해 보기 위해 다사테크의 인포테인먼트 로봇에 본 저장소를 실험해 보았다[3]. 로봇은 세 개의 외부 저장소를 이용할 수 있다는 가정하에 수행된 이 실험을 통해, 상황에 따라 필요한 컴포넌트들을 외부 저장소들로부터 획득하여 이용하게 해 줄 수 있다는 것을 확인하였다. 그러나 이 실험을 통해 두 가지 문제점을 발견하였다. 첫째, 로봇이 일일이 외부 저장소들을 접근하여야 하고 각 저장소에 저장된 모든 컴포넌트 정보를 검색해 보아야 하므로 외부로부터의 컴포넌트 획득 성능이 낮다는 것이다. 둘째, 로봇 커뮤니티의 발달에 따라 가용한 컴포넌트 저장소의 수가 많아질 것으로 예상되는데, 어떻게 로봇의 기능 획득 성능 저하 없이 저장소를 확장해 나갈 것인가와 분산된 컴포넌트들을 효율적으로 공유할 것인가 하는 문제점이다.

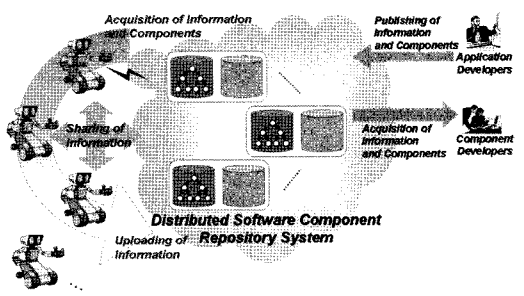

그림 1 저장소의 역할

이러한 문제점들을 해결하기 위해 본 연구에서는 분산된 외부 컴포넌트 저장소들을 컴포넌트의 기능적인 측면에 따라 논리적인 그룹으로 묶은 셀이라는 개념을 도출하였고, 이렇게 그룹화된 셀 내부의 컴포넌트들은 의미론적 웹의 한 접근법인 온톨로지를 기반으로 명세된다[2]. 본 논문에서는 이 셀기반의 접근법과 저장소 아키텍처, 프로토타입을 제시한다. 셀 기반 저장소의 프로토타입을 개발하여 실험한 결과, 셀 기반의 저장소를 이용하여 로봇이 분산된 저장소를 일일이 물리적으로 접근하는 것이 아니라 논리적으로 투명하게 접근을 할 수 있도록 지원한다. 또한, 로봇 컴포넌트/애플리케이션 개발자들이 자신의 접근 가능한 저장소를 변경하면 전체 저장소 시스템에 반영되어 다른 개발자들이나 로봇이 사용할 수 있게 해 주는 컴포넌트의 투명한 공유를 지원 한다.

2절에서는 자가 성장 소프트웨어를 지원하기 위해 필요한 요구조건들을 기술하고, 3절에서 셀의 개념을 설명한 후 4절에서 셀 기반의 저장소 구조 대해 자세히 설명한다. 5절에서는 셀 기반 저장소의 프로토타입을 기술하고 6절에서는 관련 연구들을, 7절에서는 셀 기반 저장소의 평가에 대해 기술한다. 마지막으로 8절에서는 결론과 향후 연구에 대해 설명한다.

2. 자가 성장 소프트웨어 지원을 위한 저장소의 요구조건

이 절에서는 컴포넌트 저장소의 일반적인 요구조건[4]과 자가 성장 소프트웨어를 지원하기 위한 저장소의 요구 조건에 대해 기술한다.

2.1 컴포넌트 저장소의 일반적 요구조건

본 절에서는 컴포넌트 기반의 소프트웨어 개발을 지원해 주기 위해 컴포넌트 저장소가 일반적으로 갖추어야 할 요구 조건들에 대해 기술한다. 컴포넌트 저장소는 개발된 컴포넌트를 명세할 수 있도록 지원해 주어야 하고, 컴포넌트를 체계적으로 분류하여 관리할 수 있도록 해 주어야 한다. 새로운 컴포넌트나 애플리케이션을 개발 시, 개발자들이 재사용 가능한 컴포넌트들을 검색 및 획득할 수 있는 방법을 제공해야 하며, 새로운 컴포넌트들이 개발되고 나면 이를 등록하여 다른 개발자들이 재사용 가능하게 해 주어야 한다. 또, 개발자들이 이용하는 저장소에 컴포넌트의 버전, 수정, 삭제, 업데이트 등의 변화가 발생 했을 때 이를 모니터링 해주는 방법을 제공해야 한다. 개발자들에게 저장소를 쉽게 이용할 수

있도록 사용자 인터페이스를 제공해야 하고, 저장소의 확장 및 분산이 큰 비용을 들이지 않고 수행될 수 있도록 지원해 주어야 하며, 저장소에 문제가 발생했을 때 이를 해결할 수 있는 방법을 제공해야 한다.

2.2. 자가 성장 로봇 소프트웨어를 지원하기 위한 저장소의 요구조건

본 절에서는 앞서 설명한 저장소의 일반적인 요구 조건 외에 로봇 도메인에서 자가 성장 로봇 소프트웨어를 지원하기 위해 추가적으로 만족해야 하는 요구 조건들에 대해 기술한다.

- **저장소의 진화성**: 저장소의 진화성이란 저장소가 점차 필요한 기능들의 조합을 갖출 수 있도록 변화해 가는 것을 말한다. 로봇이 배치될 환경과 접할 수 있는 상황을 모두 예측하여 필요한 모든 기능들을 로봇에게 부여하기는 어려운 일이다. 또한 로봇의 내부 자원은 무한 한 것이 아니라 제약을 가지고 있기 때문에 로봇이 처음 출시 될 때에는 로봇을 구동하는 필수적인 기능들만 가지고 있고, 자신의 환경과 상황에 따라 점진적으로 기능을 진화해 가는 것이 필요하다. 가용한 컴포넌트들이 많아짐에 따라 로봇은 여러 분산된 저장소를 이용할 수 있게 되는데, 로봇이 필요한 기능을 증식할 때 검색 및 획득의 성능을 저하시키지 않도록 진화적으로 분산 구조를 변화시킬 수 있는 저장소 체계가 필요하다.
- **투명한 저장소의 접근**: 저장소로의 투명한 접근이란, 분산된 저장소들의 구조 및 위치 등을 고려하여 일일이 접근하지 않고도 필요한 컴포넌트를 획득할 수 있도록 해주는 것을 말한다. 로봇이 분산되어 있는 저장소들을 일일이 접근하여 필요한 기능을 검색하는 것과 그 저장소들의 위치와 저장하고 있는 정보들을 유지해야 하는 것은 로봇의 성능을 저하시킬 수 있다. 따라서 저장소는 로봇이 외부 저장소들에 대한 정보는 유지할 필요 없이 요청만하면 분산된 저장소를 접근하여 필요한 컴포넌트를 획득 할 수 있게 지원해 주어야 한다.
- **투명한 컴포넌트의 공유**: 로봇 컴포넌트 개발자는 자신의 접근 가능한 저장소에 컴포넌트를 개발하여 저장하거나 업데이트 하게 된다. 새로운 컴포넌트를 개발하거나 기존 컴포넌트를 변경했을 때, 자신이 개발한 컴포넌트가 어디에 어떻게 사용될지 판단하여 연관된 분산 저장소들에 등록 및 업데이트하여 주기는 어렵다. 따라서, 개발자들이 자신이 접근 가능한 저장소에 컴포넌트를 등록하거나 변경하면 전체 분산 저장소 시스템에 바로 반영되어 로봇이나 다른 개발자들이 이용할 수 있도록 해주어야 한다. 또한 로봇은 인간과 달리 외부로부터 획득된 컴포넌트들이 자신이 이용하는 저장소의 어느 부분에 분류되어 저장되어야 하는지 판단하기 어렵다. 따라서, 이렇게 공유되는 저장소들 중에서 로봇이 필요한 컴포넌트를 검색하여 획득하였을 경우 이를 로봇이 사용하는 저장소에 자동적으로 분류하여 줄 수 있는 기능이 필요하다.
- **의미론적 검색**: 일반적으로 기존의 컴포넌트 저장소들은 키워드 기반의 검색 방법을 이용하고 있다. 그러나 이 방식은 부정확한 컴포넌트들의 검색을 초래할 수 있다[5]. 로봇에게 부정확한 컴포넌트 들이 제공된다면 로봇이 필요한 컴포넌트를 찾는데 더 많은 시간 비용을 소비해야 한다. 따라서 저장소는 가용한 많은 컴포넌트 들 중에서 다양한 속성들을 고려하여, 처한 상황에 적절하고 정확한 컴포넌트를 검색할 수 있도록 지원해주는 의미론적 검색을 지원해야 한다.

3. 셀의 개념

본 절에서는 앞서 설명한 저장소의 진화성, 투명한 접근 및 공유, 의미론적 검색의 요구조건들을 만족시키기 위해 본 연구에서 개발중인, 기능적인 측면으로 분류한 컴포넌트의 논리적인 그룹인 셀의 개념에 관해 기술하고, 셀 기반의 접근법에 대해 설명한다.

3.1 셀의 정의

셀이란 컴포넌트의 기능적인 측면을 기준으로 분산된 저장소들을 논리적으로 그룹화한 단위를 말한다. 따라서 셀은 유사한 기능을 수행하는 컴포넌트들끼리 모여서 그룹을 이루게 되게 된다. 또 이렇게 정의된 셀은 컴포넌트 개발자들이 컴포넌트를 개발함에 있어 점차 연관된 컴포넌트들을 셀이 포함할 수 있도록 진화적으로 변화해 가는 단위를 말한다.

이러한 셀 들을 생성하기 위해 본 연구에서는 서비스 로봇 도메인을 분석하여 로봇에게 꼭 필요한 일반적인 기능들을 추출하였다. 그림 2에서 볼 수 있듯이 이 기능들은 HRI(Human Robot Interaction), Communications, Locomotion, Sensing, Operation, Home Security, Power Management이다. HRI 셀은 음성인식, 얼굴인식 등의 로봇과 인간의 상호작용을 위한 컴포넌트들로 구성되어 있고, Communications 셀은 서비스 로봇이 통신을 할 때 이용하는 프로토콜 등을 제공하는 컴포넌트들로 구성되어 있다. Locomotion 셀은 로봇이 이동하는데 필

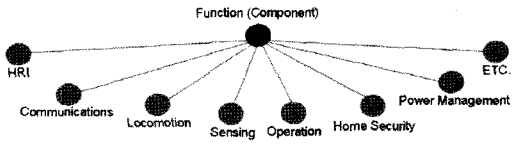

그림 2 셀의 기본 구조

요한 컴포넌트들을, Sensing셀은 로봇이 이용하는 레이저, 비전 등의 센서를 동작하는데 필요한 컴포넌트를 그룹화 하고 있다. Operation 셀은 로봇의 머리, 팔, 다리 등을 움직이는데 필요한 기능들을 담고 있고, Home Security 셀은 보안관련 컴포넌트들을, Power Management 셀은 로봇의 전원을 관리해 주는 기능을 포함하고 있다. ETC. 셀은 추후에 개발될 컴포넌트들이 이 그룹들 중에 속하지 않게 되는 것을 고려하여 추가적으로 생성된 그룹이다. 이 그룹은 컴포넌트가 다양해짐에 따라 다른 여러 그룹이 생성될 수 있음을 의미한다.

3.2 셀과 분산된 물리적 저장소들의 관계

그림 3은 물리적으로 분산된 컴포넌트 저장소들이 논리적으로 어떻게 그룹화 되는 지를 보여주고 있다. 분산된 컴포넌트들은 앞 절에서 설명한 기능적 측면에서 유사한 컴포넌트들끼리 하나의 논리적 그룹을 생성한다. 그리고 이 셀 내부에 있는 컴포넌트들에 대한 컴포넌트의 명세 즉, 컴포넌트의 온톨로지는 셀관리기에서 통합하여 관리 된다. 중앙관리기는 모든 셀들의 위치 정보를 담고 있고, 로봇과 개발자들에게 분산된 외부 저장소로의 투명한 접근을 위한 접근점을 제공한다.

유사한 기능을 하는 컴포넌트들끼리 그룹화 하면 한 저장소 내의 컴포넌트들이 여러 셀에 논리적으로 그룹화 될 수도 있다. 그림 3에서 물리적 계층의 중첩된 부분이 이러한 현상을 나타낸다. 하나의 저장소에는 개발자들이 이용하는 다양한 컴포넌트들이 존재할 수 있다. 이렇게 다양한 컴포넌트들을 물리적으로 그룹화 하지 않고 논리적으로 기능에 따라 그룹화함으로써 로봇이나 개발자들에게 필요한 기능만 찾을 수 있도록 지원해 줄 수 있다.

이러한 논리적 중첩을 허용함으로써, 어떤 셀관리기에 문제가 발생 했을 때 다른 기능을 하는 컴포넌트의 획득에는 영향을 주지 않을 수 있게 된다. 만일 중첩을 허용하지 않고 그룹화 한다면 셀관리기에 문제가 발생했을 때 다른 기능을 수행하는 컴포넌트 또한 찾지 못하게 된다. 그리고 중앙관리기와 셀관리기는 논리적 단위이고 명세 수준의 정보만 가지고 있으면 되므로 저장공간 낭비나 별도의 노력 없이 분산된 저장소들 중 어느 저장소라도 그 역할을 수행할 수 있다. 즉 외부의 여러 저장소들에게 중앙관리기와 셀관리기의 기능을 부여하여 접근하고자 하는 관리기에 문제가 발생했을 때 근접한 관리기를 접근함으로써 필요한 컴포넌트를 획득 할 수 있다.

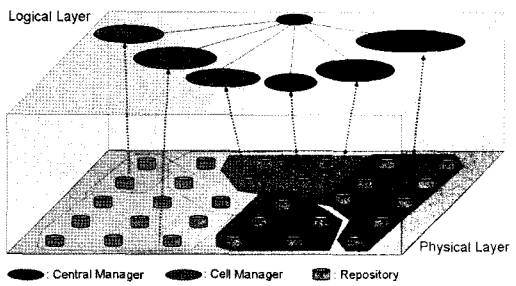

그림 3 논리적 셀과 분산된 저장소들의 관계

3.3 연관성에 기반한 셀의 변화

컴포넌트들이 점진적으로 개발되어 가용한 컴포넌트들이 많아짐에 따라 셀은 그 내부 구조가 유사한 기능을 가진 컴포넌트의 구조로 점차 진화해 나가게 된다. 또한 셀은 유사한 기능을 하는 그룹뿐만 아니라 서로 연관된 컴포넌트들끼리도 그룹화되어 진화해 나간다.

그림 4에서 한 개발자가 Sensing 셀에서 Vision Sensing 컴포넌트와 Locomotion 셀에서 Path Planner컴포넌트를 획득 후 재사용하여 Vision-based Path planner라는 컴포넌트를 개발하였다고 가정하자. 이렇게 개발된 새로운 컴포넌트는 Sensing 셀과 Locomotion 셀에 중첩되어 그룹화 된다. 이렇게 서로 연관된 컴포넌트들이 그룹화 됨으로써 로봇과 개발자들은 필요한 컴포넌트 검색 시 기능적 측면뿐만 아니라 서로 연관된 컴포넌트들까지 검색 할 수 있게 되어 서로 관련이 있는 다양한 컴포넌트들을 이용할 수 있게 된다. 이렇게 재사용되어 개발되는 새로운 컴포넌트들이 서로 연관을 가지면서 저장됨으로써 셀들은 점점 관련된 컴포넌트들끼리 그룹화 되어 진화하게 된다.

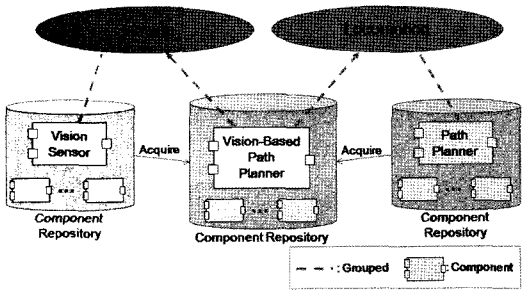

그림 4 셀의 연관기반 변화

3.4 셀 기반 접근법의 필요성

셀 기반의 접근법은 로봇이 분산 컴포넌트 저장소들을 일일이 접근하는 기존 실험에서의 문제점을 해결하기 위해 중앙관리기를 제공하여 분산된 외부 저장소들을 접근하기 위한 일정한 접근 포인트를 지원해 준다. 그리고, 로봇이 외부의 컴포넌트를 필요로 할 때 로봇은 기능 측면에서 검색을 요청하게 되는데, 유사한 기능을 하는 컴포넌트들만 검색할 수 있도록 지원해 주고, 명세

수준의 온톨로지를 검색할 수 있도록 지원해 주므로 외부 컴포넌트의 검색 성능을 향상시킬 수 있다. 뿐만 아니라 개발자들에게는 재사용 가능성이 높은 유사한 기능을 하는 컴포넌트들 컴포넌트들을 검색 할 수 있도록 지원해 주고 검색의 성능 또한 향상시켜 준다. 또한 셀은 외부 컴포넌트들의 변화에 따라 점차 서로 유사한 기능을 하거나 서로 관련 있는 컴포넌트들끼리 그룹화 되어 진화적으로 그 구조를 변화시켜 주는 단위로 이용되므로 외부 컴포넌트의 생성 및 변화를 반영하여 로봇이나 개발자들에게 이용 가능하게 해 주는 단위를 제공한다. 종합적으로 셀은 분산된 외부 저장소들의 투명한 접근과 컴포넌트의 투명한 공유를 가능하게 해준다.

4. 셀 기반의 저장소 구조

본 절에서는 셀 기반의 접근법을 이용해 구성된 저장소의 아키텍처와 그 구성요소들을 기술하고, 로봇이 셀 기반의 저장소를 어떠한 절차로 이용하는지 설명한다.

4.1 저장소의 구성요소

셀 기반의 저장소 아키텍처를 구성하는 요소들은 크게 세가지로 분류할 수 있다. 로봇 내/외부 공통요소, 로봇 내부의 요소, 그리고 로봇 외부의 요소가 그것이다.

- 내/외부 공통 요소
  - 온톨로지 저장소: 본 연구에서는 의미론적 검색을 지원하기 위해 온톨로지 기반의 접근법을 이용하고 있다. 온톨로지 저장소는 그래프 형태인 온톨로지로 기술된 컴포넌트 명세를 데이타베이스에 저장하고 있다. 이 컴포넌트의 명세는 정확한 검색을 위해 컴포넌트의 기능, 입/출력, 속성, 품질요소, 소모자원 등을 포함하고 있다.
  - 컴포넌트 저장소: 컴포넌트 저장소는 명세된 컴포넌트 들이 실제 파일 시스템의 어느 위치에 저장되어 있는지 등의 정보를 저장하는 데이타베이스와 실제 파일 저장 공간으로 이루어져 있다.
- 로봇 내부 요소
  - 컴포넌트 획득 엔진: 컴포넌트 획득 엔진은 로봇이 내부에 없는 컴포넌트를 요청할 때 이 요청을 받아 중앙관리기에 전달하고, 지정된 셀관리기로부터 추출된 컴포넌트의 후보들(온톨로지 서브셋)을 내부 온톨로지 저장소에 획득하여 주는 역할을 한다. 또한, 실제 컴포넌트 파일들을 다운로드 해주는 역할도 한다.
  - 컴포넌트 퇴거 플래너(Component Retirement Planner): 로봇이 내부 저장소에 환경과 상황에 맞는 꼭 필요한 컴포넌트만 가지고 있을 수 있도록 해 주는 요소로써 일정기간 사용되지 않거나, 자주 사용되지 않는 컴포넌트가 있을 때, 이를 내부 저장소로부터 퇴거해 주는 역할을 한다. 또한, 로봇 내부 저장소는 용량의 한계가 있어 외부 컴포넌트를 무한하게 획득할 수가 없으므로 이러한 경우 컴포넌트를 퇴거해 주는 역할도 한다. 퇴거된 컴포넌트 들은 다시 사용될 가능성을 고려하여 제거된 컴포넌트의 URIs (Uniform Resource Identifiers)를 내부 컴포넌트 저장소에 저장하여 캐쉬와 유사하게 필요시 바로 획득하여 이용할 수 있도록 지원해 준다.
  - 컴포넌트 자동 분류기: 컴포넌트 자동 분류기는 외부로부터 획득된 온톨로지의 서브셋을 내부 온톨로지 저장소의 적절한 위치에 자동적으로 병합해 주고, 그에 따라 컴포넌트 파일을 파일 시스템의 적절한 위치에 저장해 주는 역할을 한다.
- 로봇 외부 요소
  - 중앙 관리기: 중앙관리기는 분산된 외부 저장소들로의 투명한 접근점을 제공하고 셀관리기들의 위치 정보와 셀이 어떤 컴포넌트 명세들을 포함하고 있는지의 정보를 온톨로지의 상위 스키마 레벨로 가지고 있다. 로봇이 외부 컴포넌트 획득 요청 시, 중앙관리기는 이 컴포넌트가 어떠한 셀에 포함되어 있는지 파악하여 셀의 위치를 컴포넌트 획득 엔진에 통보해주는 역할을 한다.
  - 셀 관리기: 셀관리기는 셀 내부에 있는 분산된 컴포넌트 저장소들이 저장하고 있는 컴포넌트들의 전체 온톨로지를 저장하고 있다. 셀관리기는 셀 내부의 저장소들의 온톨로지 변화를 즉각적으로 반영하여 점차 서로 유사한 기능을 하거나 관련된 컴포넌트들의 그룹으로 그 구조가 진화해 갈 수 있도록 지원해 준다.
  - 브로커링 엔진: 브로커링 엔진은 중앙관리기와 셀관리기의 내부에 위치하여 중앙관리기에서는 컴포넌트의 획득 요청을 받으면 해당 컴포넌트가 저장된 셀을 선택하여 주고, 셀관리기에서는 의미론적인 검색을 기반으로 가용한 컴포넌트의 후보들을 온톨로지로부터 추출하여 주는 역할을 한다[3].
  - 저장소 관리 도구: 저장소 관리 도구는 개발자들에게 컴포넌트를 검색해 주고, 개발된 컴포넌트를 등록할 수 있도록 지원해 주며, 온톨로지를 편집할 수 있도록 지원해 주는 툴이다. 이러한 일들의 편의성을 위해 그래픽 유저 인터페이스를 제공한다.

4.2 저장소 아키텍처

그림 5는 저장소 요소들로 구성된 아키텍처와 요소들 간의 연결을 보여준다. 로봇 저장소의 요소들에 있어서 컴포넌트 획득 엔진은 컴포넌트 브로커와 서로 연결되어 있어서 외부 컴포넌트의 획득 요청을 받고 획득 후 이를 통보해 준다. 또, 컴포넌트 획득 엔진은 자동분류기를 통해 내부 온톨로지/컴포넌트 저장소에 연결되어 있다. 컴포넌트 퇴거 플래너는 온톨로지 저장소와 컴포

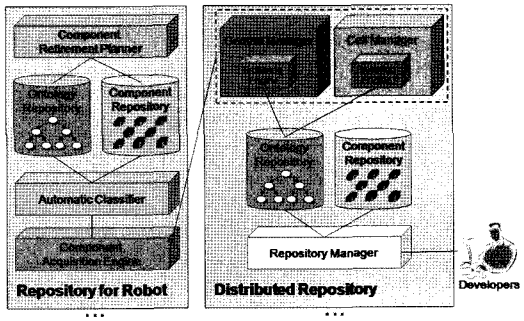

그림 5 로봇 저장소 / 분산 저장소 아키텍처

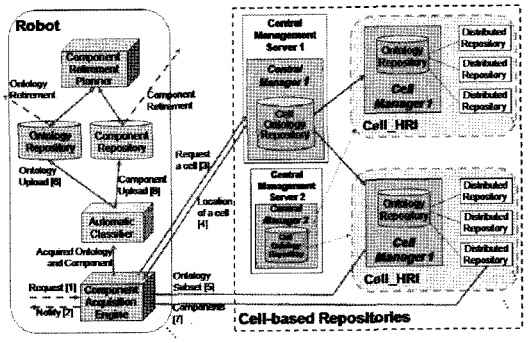

그림 6 로봇의 컴포넌트 획득 프로세스

넌트 저장소에 연결되어 내부 컴포넌트의 변화를 이 저장소들에 업데이트 한다.

분산 저장소 요소들에 있어서 중앙관리기는 각 셀관리기와 연결되어 있고 셀관리기는 온톨로지 저장소 및 컴포넌트 저장소와 저장소 관리기를 통하여 연결되어 있다. 점선 내부에 위치한 중앙관리기와 셀관리기는 서버 형태의 지정된 한 저장소 내부에 위치하는 것이 아니라 분산된 저장소들 중 할당되어 일종의 서버 역할을 하게 하는 것이다. 즉, 분산된 저장소들 중 어느 것이나 중앙관리기 및 셀관리기의 역할을 수행할 수 있다. 이렇게 중앙관리기와 셀관리기를 하나의 서버 형태가 아닌 분산 저장소에 할당함으로써 중앙 집중형 서버의 신뢰성 및 확장성 문제를 해결할 수 있다.

4.3 내부 저장소의 접근

로봇은 내부에 저장소 구성 요소 외에 온톨로지 저장소로부터 컴포넌트를 추출해 주기 위한 요소인 컴포넌트 브로커와 경험을 바탕으로 최적화된 컴포넌트를 학습엔진을 포함하고 있다[5]. 컴포넌트 브로커는 의미론적 검색에 기반하여 내부 온톨로지 저장소로부터 필요한 기능을 가진 컴포넌트의 후보들을 추출하여 주고, 학습엔진은 저장되어 있는 경험정보를 바탕으로 추출된 후보들 중에서 최적화된 컴포넌트를 선택하여 준다.

4.4 셀 기반의 분산 저장소의 접근

그림 6은 로봇이 셀기반의 분산된 저장소들로부터 컴포넌트를 획득하는 과정을 보여주고, 숫자들은 셀 기반 저장소로의 접근 순서를 나타낸다. 로봇에게 필요한 기능이 내부 저장소에 없을 경우, 다음과 같은 프로세스를 수행한다.

1) 컴포넌트 브로커는 필요한 기능의 온톨로지 스키마와 필요자원, 품질 속성 등을 포함한 정보들을 전달하며 컴포넌트 획득 엔진에게 외부 컴포넌트의 획득을 요청한다.
2) 요청을 받은 컴포넌트 획득 엔진은 접근점인 중앙관리기에 접속하여 컴포넌트 브로커로부터 받은 필요한 기능의 정보를 전달한다.
3) 중앙관리기는 내재된 브로커링 엔진을 이용하여 필요한 컴포넌트가 어떤 셀에 포함되어 있는지 식별하고 그 셀관리기의 위치를 URL(Uniform Resource Locator)로 컴포넌트 획득 엔진에게 알려준다.
4) 컴포넌트 획득 엔진은 해당 셀관리기에 접근하고 셀관리기는 내재된 브로커링 엔진을 이용하여 필요한 컴포넌트의 후보 즉, 온톨로지 서브셋을 추출해 낸다.
5) 컴포넌트 획득 엔진은 이 서브셋을 로봇 내부로 획득해 오고 컴포넌트 자동 분류기를 통해 내부 온톨로지 저장소에 업데이트 한다.
6) 학습엔진을 통해 최적의 컴포넌트가 선택되면 컴포넌트 획득 엔진은 실제 컴포넌트 파일이 저장된 저장소에 접근하여 컴포넌트를 다운로드 한다.
7) 다운로드 된 컴포넌트 파일은 자동분류기를 통해 적절한 파일시스템의 위치에 저장된다.

4.5 셀 내부에서의 상호작용

그림 7은 셀 내부에서 어떠한 상호 작용이 일어나는지 보여주고 있다. 컴포넌트 개발자들이 새로운 컴포넌트를 개발하거나 업데이트 했을 때, 저장소 관리 도구를 이용하여 이를 접근 가능한 저장소에 저장한다. 저장소 관리 도구의 온톨로지 구축 인터페이스를 이용하여 개발된 컴포넌트의 명세를 작성하거나 업데이트 된 컴포넌트의 명세를 변경하면, 온톨로지 관리 도구는 이 온톨

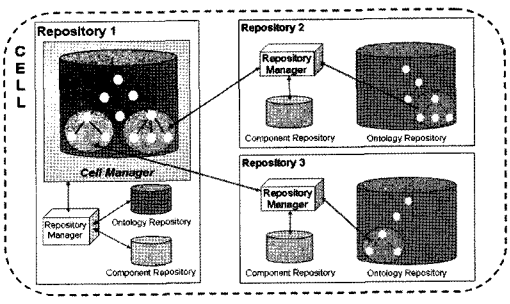

그림 7 셀 내부 상호작용

로지의 변화를 셀관리기의 온톨로지에 바로 반영하여 진화적으로 그 구조를 성장해 나간다. 이렇게 진화적으로 변화를 반영해 나감으로써 로봇이나 다른 개발자들이 즉시 이용 가능하게 해준다.

4.6 온톨로지 저장소와 컴포넌트 저장소의 연결

본 절에서는 온톨로지 저장소와 컴포넌트 저장소가 서로 어떻게 연결되는지를 설명한다. 온톨로지 저장소의 컴포넌트 인스턴스는 URIs를 가지고 있는데 이를 통해 온톨로지 저장소와 컴포넌트 저장소의 RDB(Relational Data Base) 시스템과 서로 연결되어 있다. 컴포넌트 저장소는 RDB 시스템과 실제 파일들을 저장하고 있는 파일 시스템으로 구성되어 있고, RDB에는 컴포넌트의 URIs와 실제 물리적 파일의 위치(URLs)를 저장하고 있다. 개발자들이 컴포넌트를 재 사용시 필요한 각종 명세들의 파일 내 위치를 저장하고 있고, 컴포넌트 퇴거 계획을 위한 날짜와 사용횟수를 저장하고 있다. 개발자들을 위한 명세에는 크게 문서들과 다이어그램들이 있는데 문서들은 컴포넌트의 APIs, 요구사항 명세서, 설계 문서 등을 포함한다. 다이어그램에는 UML 다이어그램들과 기타 참고할 수 있는 추상적 다이어그램들을 포함하고 있다. 컴포넌트의 검색 시 온톨로지의 URIs를 통해 로봇은 필요한 컴포넌트를 획득 할 수 있고, 개발자들은 필요한 컴포넌트와 그 컴포넌트의 명세를 획득할 수 있다.

5. 저장소 프로토타입

이 절에서는 셀 기반 컴포넌트 저장소의 프로토타입을 기술한다. UML 다이어그램들과 로봇 내부 저장소와 셀 기반의 외부 저장소들을 모니터링 할 수 있는 모니터들, 그리고 저장소 관리 툴에 대해 설명한다.

5.1 UML 모델

그림 8은 셀 기반 저장소 사용자들의 사용사례 다이어그램을 보여준다. 사용자들은 크게 로봇과 로봇 컴포넌트 개발자, 로봇 애플리케이션 개발자, 중앙관리자, 셀 관리자로 분류할 수 있다. 로봇은 내부 온톨로지 저장소를 검색하여 가용한 컴포넌트의 후보들을 추출하고, 외부 저장소들로부터 필요한 컴포넌트 획득 시 이를 분류하여 내부의 온톨로지/컴포넌트 저장소를 업데이트 한다. 그리고, 자주 사용되지 않거나 불필요한 컴포넌트들을 내부 저장소로부터 퇴거하는 일을 할 때 저장소를 이용하게 된다.

로봇 컴포넌트 개발자는 로봇에게 초기의 온톨로지를 생성해 주고, 컴포넌트 개발 시 외부 온톨로지 저장소에 명세를 업데이트 하고 실제 컴포넌트 파일들을 등록한다. 또, 새로운 컴포넌트를 개발할 때 외부 저장소들을 검색함으로써 필요한 컴포넌트들을 획득한다. 로봇 애플리케이션 개발자들은 애플리케이션을 개발할 때 컴포넌트들을 조합하여 개발하게 되는데 필요한 컴포넌트들을

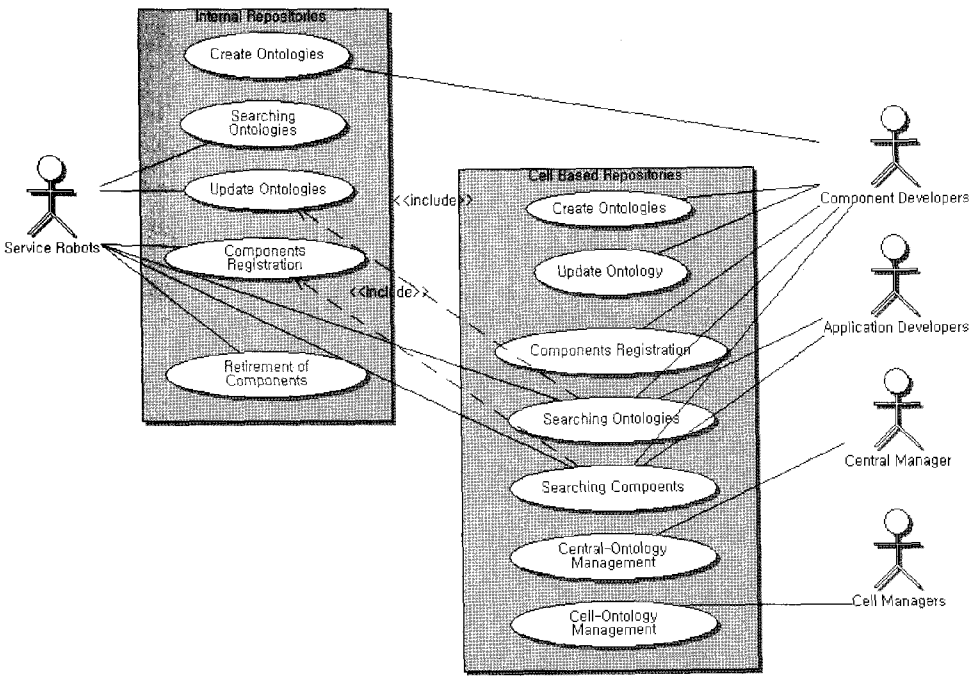

그림 8 저장소의 사용사례 다이어그램

외부 저장소들로부터 검색하고 획득하게 된다. 중앙관리자는 셀의 정보를 담고 있는 중앙관리기를 관리하는 역할을 하고, 셀관리자는 셀 내부의 컴포넌트들을 관리하고 분류하는 역할을 담당한다. 중앙관리자와 셀관리자의 역할은 셀의 진화성으로 인해 부담이 작으므로 컴포넌트 개발자나 애플리케이션 개발자가 그 역할을 할 수 있다.

5.2 저장소 프로토타입

본 연구에서 저장소를 구현하기 위해 윈도우 XP와 리눅스를 이용하였고, 자바 프로그래밍 언어를 이용하였다. 온톨로지를 구축, 편집하기 위해 스탠포드 대학에서 개발된 Protégé 툴을 이용하였고, Protégé를 이용하여 RDF/RDFS 파일을 생성하였다[6,7]. 이렇게 생성된 온톨로지를 온톨로지 저장소 형태로 저장하기 위해 Jena2 라이브러리와 My-SQL 서버를 이용하였다[8]. Jena2 라이브러리는 RDF/RDFS 파일 접근을 위한 라이브러리를 제공하여 RDF 트리플 구조로 RDB에 저장할 수 있도록 지원해 준다. My-SQL 서버를 자바에서 접근하기 위하여 JDBC 드라이버를 이용하였다. 외부로부터의 온톨로지 획득을 위해 HTTP를 이용하였고, 실제 컴포넌트 파일의 다운로드를 위하여 FTP 서버를 이용하였다.

5.3 저장소 모니터

그림 9는 로봇의 내/외부 저장소를 모니터링 하기 위한 GUI들을 보여준다. 모니터는 크게 로봇 내부의 저장소 상태를 보여주는 로봇 내부 모니터와 로봇 외부의 셀 기반 저장소의 상태를 보여주는 셀 기반 저장소 모니터로 구분 할 수 있다. 로봇 내부 모니터는 현재 로봇이 가지고 있는 컴포넌트 파일의 리스트를 보여주고, 박스로 표시된 부분이 외부로부터 획득된 새로운 컴포넌트를 보여준다. 그림에서 로봇이 비전 기반의 패스플래너를 획득 하였다는 것을 보여준다. 셀 기반 저장소 모니터는 중앙관리기, 셀관리기, 실제 컴포넌트 저장소를

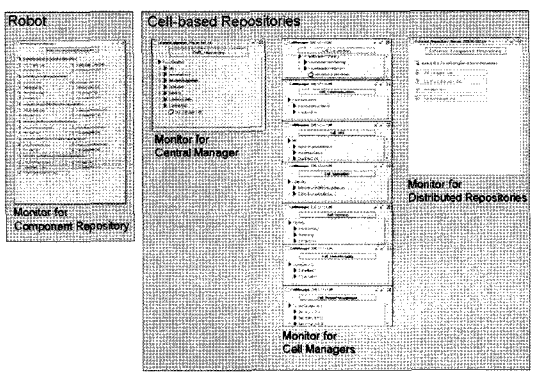

그림 9 로봇 내부 / 셀 저장소 모니터

포함하고 있고, 이 그림은 비전 기반의 패스플래너를 요청 받아 Locomotion 셀을 알려주고 그 셀 내부에서 최종적으로 비전 기반의 패스플래너 컴포넌트가 검색되어 로봇에게 획득 할 수 있도록 지원해주는 상태를 보여주고 있다.

5.4 저장소 관리 툴

그림 10은 컴포넌트 개발자들이 컴포넌트를 명세, 등록하도록 지원해 주는 저장소 관리 툴을 보여준다.

이 툴에서 왼쪽 부분은 현재 개발자 자신의 저장소에 저장되어 있는 컴포넌트들의 온톨로지 명세를 보여주고, 오른쪽 상단 부분은 컴포넌트의 명세 시 입력해야 할 항목들을 보여준다. 현재 본 연구에서는 컴포넌트의 명세를 위해서 기능, 자원소모량(CPU. Memory, Bandwidth)을 고려하고 있다. 이는 추후 사용자들의 품질 속성까지 고려한 명세로 확장될 전망이다. 툴의 오른쪽 하단 부분은 새로 개발된 컴포넌트를 등록하기 위한 인터페이스를 제공한다.

컴포넌트 개발자가 새로운 컴포넌트를 개발했을 때, 개발자는 자신이 개발한 컴포넌트가 온톨로지 구조의 어느 부분에 분류되어야 하는지 온톨로지 명세 인터페이스를 통해 판단한다. 이렇게 자신의 컴포넌트를 분류한 후에 오른쪽 상단의 Description 인터페이스를 이용해 기능과 자원 소모량을 기술하면 내부 온톨로지 저장소에 기술한 명세가 업데이트 된다. 마지막으로 실제 개발된 컴포넌트 파일을 오른쪽 하단의 Registration인터페이스를 이용하여 자신의 로컬 파일 시스템에 저장하면 온톨로지 명세는 개발자가 속한 셀관리기에 업데이트 되어 등록된다.

그림 10은 현재 개발자가 비전기반의 패스플래너를 개발한 뒤 패스플래너 온톨로지의 하위에 자신이 개발한 컴포넌트를 분류하고 있는 것을 보여준다. 분류한 후에 비전기반 패스플래너 컴포넌트의 기능을 "Vision-BasedPath Planner"라고 기술하고, 예상 CPU 소모량을 10%, 메모리 소모량을 15%, 대역폭을 5.67Mhz로 기술한 것을 보여주고 있다. 또, 자신의 로컬 컴포넌트 저장소의 파일 시스템에 비전 기반 패스플래너 컴포넌트를 등록하는 것을 보여준다. 이렇게 등록을 거치면 Sensing셀과 Locomotion셀의 셀관리기에 업데이트 된 컴포넌트의 온톨로지 명세가 자동으로 반영되어 로봇이나 다른 개발자가 이용할 수 있도록 해준다.

6. 관련 연구

로봇 커뮤니티의 발달에 따라 많은 가용한 컴포넌트와 저장소를 다루기 위한 본 연구는 분산 온톨로지 저장소, 분산 컴포넌트 저장소와 많은 관련이 있다. 본 절에서는 이 두 가지 관련 연구에 대해 기술한다.

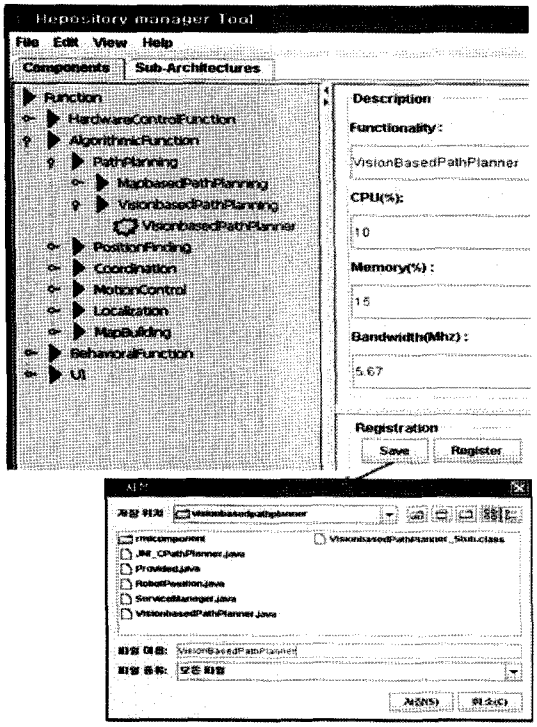

그림 10 컴포넌트 개발자를 위한 저장소 관리 툴

6.1 분산 온톨로지 저장소

현재 분산 온톨로지 저장소 관련하여 많은 연구가 이루어지고 있다. 이 저장소들은 온톨로지 기반의 접근법을 이용하여 분산된 지식(온톨로지)의 효율적인 공유와 관리를 목적으로 하고 있다. 표 1은 이 저장소들을 종합하여 보여주고 있다.

기존의 분산 온톨로지 저장소들은 의미론적인 검색은 지원하고 있지만 찾고자 하는 기능과 연관된 온톨로지의 부분적 검색의 미비, 저장소 서버에 문제가 발생하였을 때 이를 대처할 수 있는 방법의 부족, 분산된 저장소들로의 투명한 접근 방법 부족, 고정된 분산 저장소의 구조로 인한 유동성 부족, 분산의 기준 불분명 등의 문제점을 가지고 있다. 따라서, 실시간 기능획득과 필요한 컴포넌트들과 관련 컴포넌트들 만을 유지해야 하는 로봇 도메인에서 이용하기는 아직 부족하다[19].

표 1 분산 온톨로지 저장소

	DOMS [9]	OnBrowser [10]	SesameRDF [11]	RDFPeers [12]
Searching Mechanism	Semantic	Semantic	Semantic	Semantic
Fault-tolerance of Repository	Not support	Support	Not support	Support
Transparent Access	Not support	Not support	Support	Not support
Transparent Sharing	Not support	Not support	Not support	Not support
Evolutionary Support	Not support	Not support	Not support	Not support
Criteria for Scalability	Unclear	Unclear	Unclear	RDF Triple

6.2 분산 컴포넌트 저장소

분산 컴포넌트 저장소 관련 연구들도 많이 찾아볼 수 있는데 컴포넌트 기반의 소프트웨어 개발을 지원하기 위해 주로 라이브러리 접근 법을 이용하여 점차 많아지는 가용한 컴포넌트들의 효율적인 재사용을 그 목적으로 하고 있다. 표 2는 분산 컴포넌트 저장소들을 종합하여 보여주고 있다. 대부분의 컴포넌트 저장소들은 컴포넌트의 검색에 있어서 키워드 기반의 검색 방법을 이용하고 있고, 그 사용 대상이 사람들이므로 자동화된 컴포넌트의 분류를 지원하지 않고 있다. 대부분 라이브러리 접근법을 이용하고 있어서 라이브러리 전체를 일일이 찾아봐야 하고, 중앙 집중 형 관리 체제이기 때문에 라

표 2 분산 컴포넌트 저장소

	CodeFinder-PEEL [13]	Agent-based Repository [14]	CKBR [15]	Mediator-based Repository [16]	Component Hub [17]	UDDI-based Repository [18]
Searching Mechanism	Keyword	Keyword	Keyword	Keyword	Keyword	Keyword
Fault-tolerance of Repository	Not support	Not support	Not support	Not support	Support	Not support
Transparent Access	Support	Not support	Support	Not support	Not support	Support
Transparent Sharing	Not support	Support	Not support	Not support	Not support	Not support
Evolutionary Support	Semi-support	Not support	Not support	Not support	Not support	Not support
Criteria for Scalability	Unclear	1 repository = 1 server	Unclear	Unclear	Unclear	Unclear

이브러리에 문제가 발생 했을 때 이에 대한 대처 방안이 부족하다. 물리적으로 분산된 컴포넌트들에 대한 투명한 접근이 부족하고, 분산 온톨로지 저장소들과 마찬가지로 고정된 저장소 구조를 가지고 있고 분산 및 확장에 대한 기준이 모호하다. CodeFilnder-PEEL 저장소 경우는 진화적인 저장소 구조를 지니고는 있지만, 코드 레벨의 검색과 구조의 진화를 위해서는 항상 사용자의 간섭이 필요하다. 이러한 기존의 컴포넌트 저장소들도 온톨로지 저장소들과 마찬가지로 로봇 도메인에서 자가 성장을 지원하기 위하여 이용되기는 아직 부족한 점들이 있다[19].

7. 평 가

본 절에서는 셀 기반의 저장소를 평가하기 위해 앞서 서술한 요구조건들을 만족 시키는지 여부에 대해 기술한다.

표 3은 앞서 서술한 컴포넌트 저장소의 요구조건들과 이 요구 조건들을 만족시키는 본 연구의 요소들을 보여주고 있다.

1) 본 저장소는 온톨로지 기반의 접근 법을 이용하여 컴포넌트를 명세 할 수 있도록 지원해 주고 이 명세를 RDB형태의 온톨로지 저장소로 저장하여 관리할 수 있도록 해준다.
2) 본 저장소는 컴포넌트 획득 엔진과 외부의 브로커링 엔진을 이용하여 외부 저장소들로부터 필요한 컴포넌트의 명세인 온톨로지와 컴포넌트 파일들을 획득해 올 수 있다.
3) 컴포넌트 퇴거 플래너는 로봇 내부의 컴포넌트의 변화를 모니터링 하고 필요 시 내부로부터 컴포넌트들을 퇴거해 주는 역할을 한다.
4) 저장소 관리 도구와 셀들은 개발된 컴포넌트들을 등록할 수 있도록 지원해 주고 등록된 컴포넌트들은 즉시 전체 저장소에 반영되어 다른 개발자들이나 로봇이 이용할 수 있도록 해준다.
5) 저장소 관리 도구는 온톨로지 구축과 컴포넌트의 등록을 지원하는 UI를 지원하여 준다.
6) 컴포넌트 퇴거 플래너는 로봇이 내부에 환경과 상황에 맞는 컴포넌트들만 유지할 수 있도록 지원해 주고, 필요 시 자주 사용되지 않는 컴포넌트 들은 로봇 내부로부터 퇴거하여 준다. 셀들은 외부 저장소들의 컴포넌트 변화에 반영하여 점차 더 유사하고 관련 있는 컴포넌트들이 그룹을 이루게 되는 진화적인 컴포넌트 저장소 구조를 지원한다.
7) 중앙관리기는 로봇이나 개발자들에게 물리적으로 분산된 저장소들로의 투명한 접근점을 지원해 주고, 셀은 실제 분산된 저장소들을 일일이 접근하는 것이 아니라 논리적인 단위로 접근 가능하게 해주므로 투명한 접근을 지원한다.
8) 개발자들이 자신들이 이용 가능한 저장소에 새로운 컴포넌트를 등록하거나 업데이트하면 셀관리기는 이러한 변화를 즉각 반영하여 다른 개발자들이나 로봇들이 이용할 수 있게 해준다.
9) 중앙관리기와 셀관리기 내부에 있는 브로커링 엔진은 다양한 요소들을 고려하여 컴포넌트의 의미론적인 검색을 지원해 준다.
10) 셀은 컴포넌트의 기능적인 분류인 그룹들을 제공하여 새로운 저장소나 컴포넌트가 추가 되어야 할 때 이러한 그룹에 속할 수 있게 되고, 논리적인 단위의 확장이 가능하기 때문에 이러한 확장에 따른 컴포넌트의 획득 성능이 저하되지 않는다.
11) 셀과 셀관리기, 중앙관리기는 컴포넌트의 명세 수준의 정보들을 이용하여 여러 개로 구성되어 있다. 따라서, 중앙관리기나 셀관리기에 문제가 발생 했을 때, 인접한 중앙관리기나 셀관리기에 요청하여 필요

표 3 저장소를 위한 요구 조건과 만족 요소

	Requirements	Repository Elements
1	Component Description Support	Ontology Repositories
2	Search and Retrieval	• Component Acquisition Engine • Brokering Engine
3	Classification	Automatic Classifier
4	Checking Changes of Components	Component Retirement Planner
5	Components Registration	Repository Manager and Cells
6	Usability	Repository Manager Tool
7	Evolvability	Component Retirement Planner and Cells
8	Transparent access	Central Manager and Cell Manager
9	Transparent sharing	Cell Manager
10	Semantically-based search	Brokering Engine
11	Scalability	Cells (Logical Groups)
12	Reliability	Cells and Cell Manager

한 컴포넌트 를 획득 할 수 있다.

8. 결론 및 향후 연구

본 연구에서는 지능형 서비스 로봇에게 자가 성장 로봇 소프트웨어를 지원하기 위한 기반 기술인 저장소 시스템을 개발하여 왔다. 로봇이 처할 수 있는 모든 환경과 상황을 예측하여 필요한 기능을 모두 로봇 내부에 가지고 있을 수 없다는 문제점을 해결하기 위해, 외부 저장소를 이용한 실시간 기능 획득을 지원해 주는 저장소 시스템을 개발하였고 실제 로봇 플랫폼에 적용 실험을 해 왔다. 이러한 실험을 통해 도출된 문제점들은 첫째, 로봇이 일일이 분산된 외부 저장소들을 접근해야 하므로 외부 컴포넌트 획득 성능이 낮다는 것이다. 둘째, 로봇 커뮤니티의 발달로 인해 가용한 분산 저장소들의 수가 많아짐에 따라, 로봇의 획득 성능을 유지하면서 어떻게 저장소를 확장하고 컴포넌트들을 공유할 것인가 하는 문제점이다.

본 연구에서는 이러한 문제점들을 해결하기 위해 컴포넌트의 기능적인 측면을 이용하여 물리적으로 분산된 저장소들을 논리적으로 그룹화한 단위인 셀을 정의하였고, 셀을 나누는 기준은 서비스 로봇 도메인에서 일반적으로 필요한 기능들이다. 이러한 셀 기반의 접근법을 이용하여 셀 기반의 아키텍처를 개발하였고, 셀 기반 저장소 구현을 위한 프로토타입을 제시하였다. 이 프로토타입을 통한 실험 결과, 셀은 로봇에게 일정한 접근점을 제공하고, 기능측면의 논리적인 접근을 지원하여 분산된 저장소들을 투명하게 접근할 수 있도록 지원해 준다는 것을 확인하였다.

셀 기반의 저장소를 이용하면 로봇과 개발자들이 전체 저장소를 검색하는 것이 아니라, 유사한 기능을 수행하는 컴포넌트들만 검색할 수 있도록 해 주므로 검색의 성능을 향상 시킬 수 있다. 또한 셀 기반 저장소는 개발자들이 자신의 접근 가능한 저장소나 로컬 저장소에 개발된 컴포넌트를 개발하여 기술해 놓으면, 자동적으로 이러한 변화가 반영되어 다른 개발자들이나 로봇이 이용할 수 있도록 해주는 투명한 공유를 지원해 준다.

가용한 저장소들이 수가 많아지는 것을 수용하기 위해, 현재 셀 개념을 확장하여 계층적인 셀들의 구조를 고안 중에 있고 이러한 계층적인 셀의 진화를 자동적으로 수행할 수 있는 방법을 개발 중에 있다. 또한 저장소 관리 도구 확장 하여 컴포넌트 개발자뿐만 아니라 로봇 애플리케이션 개발자, 저장소 관리자 들을 위한 도구 들을 개발 중에 있다. 또한, 셀 기반의 저장소를 실제 로봇 플랫폼에 적용 시키는 실험을 진행 중에 있다.

참 고 문 헌

[1] P. Lars and et al., "Towards an Intelligent Service Robot System," Proceedings of International Conference on Intelligent Autonomous Systems, 2000.

[2] Hyung-Min Koo and In-Young Ko, "A Repository Framework for Self-Growing Robot Software," Proceedings of 12th Asia-Pacific Software Engineering Conference (APSEC'2005), Taiwan, December 2005.

[3] Hyung-Min Koo and et al., "A Repository Framework for Architecture Based Self-Growing Robot Software," Proceedings of 2nd Korea Conference on Software Engineering, KCSE' 06, 2006.

[4] Michael Blaha, "Requirements for Repository Software," IEEE, 1998.

[5] Vijayan S. and et al., "A Semantic-Based Approach to Component Retrieval," ACM, SIGMIS Database, Vol.34. No.3, 2003.

[6] Shelly Powers, "Practical RDF," P.16-P.22, O'Reilly publication, July, 2003.

[7] What is Protégé?, http://protege.stanford.edu/overview/

[8] Jena Semantic Web Framework, http://jena.sourceforge.net/

[9] Robert Harrison and Christine W. Chan, "Distributed Ontology Management System," CCECE/CCGEI, Saskatoon, IEEE, 2005.

[10] Mario Cannataro and et al., "Distributed Management of Ontologies on the Grid," Proceedings of the 14th IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprise (WETICE' 05), IEEE, 2005.

[11] Gergely Adamku and Heiner Stuckenschmidt, "Implementation and Evaluation of a Distributed RDF Storage and Retrieval System," Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05), IEEE, 2005.

[12] Min Cai and Martin Frank, "RDFPeers: A Scalable Distributed RDF Repository based on A Structured Peer-to-Peer Network," WWW2004, ACM, 2004.

[13] Scott Henninger, "Supporting the Construction and Evolution of Component Repositories," Proceedings of ICSE, IEEE, 1996.

[14] Seong-Jae Won and et al., "A Search Agent System for Distributed Component Repository," Proceedings of the 32nd KISS Fall Conference, Seoul, November 2005.

[15] Padmal Vitharana et al., "Knowledge-Based Repository Scheme for Storing and Retrieving Business Components: A Theoretical Design and an Empirical Analysis," IEEE transactions on Software Engineering, Vol.29, No.7, July 2003.

[16] Regina M. M. Braga et al., "The Use of Medi-

ation and Ontology Technologies for Software Component Information Retrieval," ACM, 2001.
[17] James X. Ci and Wei-Tek Tsai, "Distributed Component Hub for Reusable Software Components Management," Computer Software and Application Conference, COMSAC 2000, IEEE, 2000.
[18] Dong-Keun Lee and Eun-Man Choi, "A Study on Integrating UDDI Registry and Web-Based Component Repository," Proceedings of the 31st KISS Fall Conference, Vol.31, No.2, 2004.
[19] Hyung-Min Koo, "Cell-based Approach for Evolutionary Component Repositories for Intelligent Service Robots," Mater Thesis, Information and Communications University, 2007.

구 형 민

2004년 금오공과대학교 컴퓨터공학과(학사). 2007년 한국정보통신대학교 컴퓨터공학과(석사). 2007년 한국전자통신 연구원 위촉연구원. 2007년~현재 한국정보통신대학교 컴퓨터공학과 박사과정. 관심분야는 자가적응 소프트웨어, 그룹 인지 소프트웨어, 사용자 중심 소프트웨어

고 인 영

1990년 서강대학교 전산학과(학사). 1992년 서강대학교 전산학과(석사). 2003년 미국 Univ. of Southern California(박사). 1993년~1996년 공군사관학교 교관(전임강사). 1997년 미국 USC Information Sciences Institute(ISI) 연구조교. 2003년 미국 USC ISI Postdoctoral Research Associate. 2004년~현재 한국정보통신대학교 공학부 조교수. 관심분야는 소프트웨어공학, 웹공학