A Cell-Based Approach for Evolutionary Component Repositories for Intelligent Service Robots

Hyung-Min Koo, In-Young Ko
Information and Communications University (ICU)
119 Munjiro, Yuseong-gu, Daejeon, 305-732, Korea
{hyungminkoo, iko}@icu.ac.kr

Abstract

Self-growing software is a software system that grows its functionalities and configurations by itself according to dynamically monitored situations. A component repository system is one of the essential parts of self-growing robot software, and the SemBots project at ICU is developing a cell-based distributed repository system that reconfigures its structure dynamically for reflecting accessibility evolutionarily. To accomplish evolutionary repositories, we invent the concept of a cell that is a logical grouping of distributed repositories based on an ontology hierarchy. In addition, it is also a unit for evolutionary reconfiguration of the repository structure. 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 management of components for developers.

Keywords: Self-growing Software, Intelligent Service Robots, Distributed Ontology Repositories, Distributed Component Repositories

1. Introduction

An intelligent service robot is "a robot that senses the environment, recognizes troubled situations, determines how to solve the problems and performs relevant behaviors to overcome situations" [1]. Intelligent Service Robots can not contain all functionalities that are required in an internal system, because it is hard to anticipate all situations that robots could be faced with. However, as the robotic community grows gradually, software components are continually being updated and new components are continually being developed for this purpose.

A component repository system is one of the essential parts of self-growing robot software. Robots can add to their functionalities from various external repositories, so that the internal repositories of robots can contain essential components reflecting changes of components and recognition of an environment. This process is shown in Fig 1. Robots can share their knowledge and components with each other by using a repository system, so that the trials and errors of these robots can be reduced and the robots' self-evolution processes can be improved. Component /application developers can also share components to develop robot components and applications. Through the repository system, developers can register new robot components and applications to make robots or human users use them more effectively, and so that reusability of components can be improved.

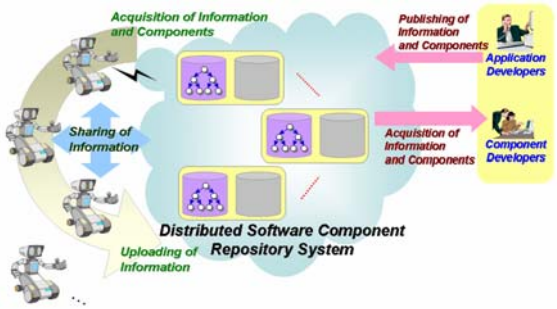

Fig. 1. **Usages of Repositories for Intelligent Service Robots**

To apply the repositories to a robot platform, an experiment was performed using the repositories on an Infotainment robot platform from DASA tech. We experimented with an initial and essential functionality that supported the acquisition of ontologies and components for robots. Fig 6 below shows screen shots of the experiment with the components of the repositories listed below them. For this experiment, three external repositories were used.

21st International Conference on
Advanced Information Networking and Applications Workshops (AINAW'07)
0-7695-2847-3/07 $20.00 © 2007 IEEE

As Fig 2 shows, the robot contains laser-based, IR-based and Sonar-based path planners in its internal repository. The robot receives a human user's command, and then it tries to go to a goal position based on a laser-based path planner. During navigation, it meets an unknown and unexpected obstacle which the laser sensor cannot detect. After that, the robot changes from that path planner to other path planners: the IR-based path planner and Sonar-based path planner of internal repository. However, the robot cannot avoid the obstacle with those sensor-based path planners. The robot makes a decision that a Vision-based path planner is needed to avoid this obstacle by decision maker & learner, and connect to external repositories. The robot finds out that the third external repository contains a Vision-based path planner and acquires that component. By changing the path planner to Vision-based path planner, the robot can avoid the obstacle and can reach the goal position.

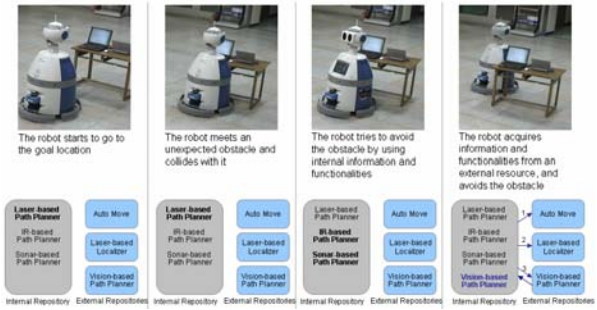

Fig. 2. **An Experiment using an Infotainment Robot**

Two more problems were defined from the experiment: low performance in the acquisition process and low manageability and sharability of external repositories.

The performance of accessing external repositories was low, because of the one-on-one connection between the robots and the external repositories. This resulted in low performance of searching and acquisition for components, because robots and developers had to connect to each external repository. Robots and developers had to know the information and data contained in the external repositories and had to search through all of the knowledge in the external repositories. Therefore, this resulted in low performance of the search process.

This experiment assumed that there were just three external repositories, but available components and repositories will grow as robot communities continue to evolve. Therefore, manageability and sharability of external repositories is low, because one repository is one server. It results in inefficient management for physically distributed repositories, and cannot support faults of external repositories. Robots and developers can not acquire new components when repositories are faced with problems.

In the final paper, we will describe the requirements and architecture of the cell-based approach for the evolutionary component repository system for self-growing robot software. Prototype implementation and an experiment with the repository system will also be presented. In Section 2, the requirements of the repository system for self-growing software will be explained. In Section 3, related work will be discussed. In Sections 4 and 5, we will explain our approaches and the prototype implementation of the repository system. In section 6, we will evaluate our approach by discerning whether our approach meets the requirements or not. Finally, the thesis will be concluded in Section 7.

2. Related Work

Our repository system is related to distributed ontology repositories and distributed component repositories. We analyze these related works briefly in this section.

2.1 Distributed Ontology Repositories

There are some researches on distributed ontology repository. These distributed ontology repositories have some limitations: the fixed structure of repositories, lack of evolutionary support, unclear criteria for distribution, lack of transparent access, inefficient management of knowledge, and lack of a reliability support. [3, 4, 5, 6]

2.2 Distributed Component Repositories

There are some researches on distributed component repository. These distributed component repositories have also some limitations to support self-growing robot software: the fixed structure of repositories, lack of evolutionary support, unclear criteria for distribution, keyword-based search, manual classification, lack of transparent access, and lack of a reliability support. [7, 8, 9, 10, 11, 12]

3. Requirements of Repositories for Intelligent Service Robots

In this subsection, we discuss specific requirements of the software component repository system for self-growing robot software.

- **Evolvability**
  - *Evolvability for robots*: There are many kinds of environments that robots can face with. Therefore, robots have to contain functionalities and

21st International Conference on
Advanced Information Networking and Applications Workshops (AINAW'07)
0-7695-2847-3/07 $20.00 © 2007 IEEE

configurations that are appropriate for each environment instead all robots have same functionalities and configurations. In addition, the repositories should have criteria for replacing components stored in internal repositories with newly acquired or updated components. These criteria are used to decide what components have to be retired and what components have to be uploaded to the internal repository.

- *Evolvability of repositories structure for robots and developers*: As a scale of distributed knowledge and the number of components are increased gradually, repositories should provide way to evolve their structure by reflecting newly updated or developed knowledge and components from distributed repositories, so that robots and developers can use those knowledge and components effectively.

- **Transparent access**: Transparent access provide a way to access various external repositories with a little overhead, in comparison with accessing each distributed repository. If robots should access each external repository, robots have to know locations of distributed repositories and data of components those are stored in the repositories, and it is hard to search appropriate components. This makes the performance of acquisition process low.

  Therefore, repositories should provide the transparent access to search external repositories.
- **Transparent sharing**: When knowledge is updated or new components are developed, component developers use their accessible repositories. Therefore, repositories should provide the transparent sharing that supports developers can just update into their accessible repositories then updated knowledge or components are reflected to robots and other developers.
- **Semantically-based search**: Usual component repositories use keyword-based indexing and searching. This keyword-based approach results in inaccurate search results [13]. Therefore, it is needed to support semantically-based matching mechanism that can retrieve components based on various aspects: functionalities, quality attributes, and properties of components (e.g. resources for components, user requirements, etc.).
- **Automatic Classification**: Human users can decide where to classify components that are acquired from external repositories. However, since robots can not make judgment, the repositories should support to automatic classification of acquired components for robots.

4. A Cell-based Repositories Architecture

In this section, we describe our approaches for the cell-based repository for self-growing robot software. We also explain the architecture of the repository.

4.1 A Cell

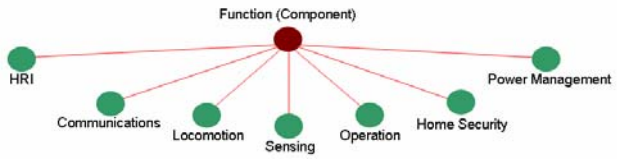

Fig. 3. **A Structure of Cells**

A ***cell*** is a logical grouping unit of distributed repositories based on functionalities of components. These groups contain components that have similar role of functionalities. As the Fig. 3 shows, components can be classified to some parts based on generalized functionalities of intelligent service robot domain: HRI (Human Robot Interaction), communications, locomotion, sensing, operation, home security, and power management. Therefore, a structure of cells consists of the highest schemas of component ontology. HRI is a group for user interface between robots and human users, such as voice recognition, face recognition. Communications is a group for data communication protocols and their application components. Locomotion is for moving robots' wheels, legs, arms, and so on. Sensing is for sensors such as laser sensors, vision sensors, sonar sensors, and so on. Home security is a group for detecting and managing criminals. Finally, power management is for managing battery power of robots. Robots can not keep connecting power line during behaviors because line makes robots' range short and small.

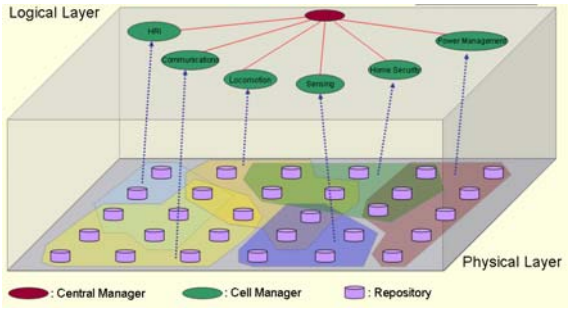

Fig. 4. **Mapping of Cells and Distributed Repositories**

Components and component repositories are distributed physically, as Fig. 4 shows. We make logical groups of components that are similar or related

to each other based on cell approach. Therefore, these components can be contained into some groups. For example, vision-based navigation components can be grouped into sensing group and locomotion group. A cell manager contains whole knowledge of a cell, and has a role for searching ontologies of a cell. A central manger contains information of cells and has a role for providing a transparent access point. When new repositories are inserted to cell-based repositories, they are contained into one or some groups, so that repositories can be extended easily. In addition, logical groups provide alternative accessing point to acquire components when an access point or a cell meets problems.

4.2 Rationales of Cells

Cells provide units for accessing physically distributed repositories transparently because cells provide logical access point and logical search for physically distributed repositories. Robots and developers can access just logical units without accessing physical repositories respectively.

Cells provide units for efficient search. In the last experiment, robots have to search for whole knowledge in a repository, and this makes a performance of searching process low. Moreover, reusability of similar and related components is higher than other components for developers. Cells provide partial search for components that are needed to robots and developers because cells are made by similar and related components.

Cells provide units for evolutionarily reflecting changes of components in physically distributed repositories. Changes of components are reflected to a cell automatically, so that robots and developers can use changed components immediately. Synthetically, cells provide transparent access and transparent sharing.

4.3 The Overall Repository Architecture

In this subsection, we describe an overall architecture of the cell-based repositories.

As Fig. 5 shows, cell-based repositories consist of two parts: internal repositories for robots and cell-based external repositories. Internal repositories consist of two parts: ontology repository and component repository. External repository system has a central manager, cell managers, cells, and repositories for developers. Central manager have connections with cells and contains hierarchy of cells. Each cell makes a logical group of repositories those are distributed physically. This distribution conforms to component ontology, so repositories in architecture are also logical. Some physical repositories can be involved in several cells because repositories can have various components those can be related to several cells.

Even though we separate the central manager, the cells, and the repositories, any distributed repositories can have a role for the central manager and the cell manager. When the fault occurs in designated central or cell manager, neighbor of it has a role for it to support reliability.

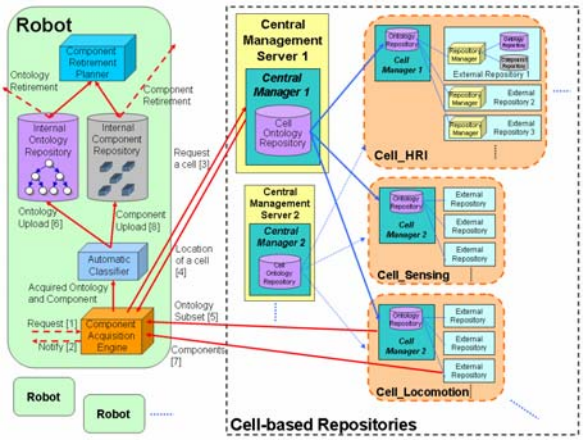

Fig. 5. **A Cell-based Repositories Architecture**

4.4 Accessing Internal Repositories

Robots have *component broker* for collecting appropriate component descriptions [4, 17]. By accessing internal repositories, the component broker searches for the components that provide functionalities to make complete the selected architecture. It extracts a set of component candidates by using a semantically-based interoperability measurement, and sends them to a learning engine. The leaning engine chooses the most appropriate component based on the history of utilizing components for a certain set of situations.

4.5 Accessing Cell-based Distributed Repositories

As Fig. 5 shows, if there is no available or suitable component in the internal repository of a robot,

Step 1) The component broker send a request to component acquisition engine.

Step 2) The component acquisition engine sends a message that contains query to central manager.

Step 3) The central manager searches for appropriate cell that contains needed component description by using semantically-based searching in cell hierarchy.

Step 4) The cell manager of selected cell searches component descriptions and extract sub-ontology from a ontology of the cell. Sub-ontology means a part of ontology that is semantically-matched with query. We

extract sub-ontology by using component schema and properties (e.g. resources) from a ontology.

Step 5) Extracted sub-ontology is acquired by component acquisition engine and it is updated in internal ontology repository by using automated classification.

Step 6) The brokers searches updated internal ontology repository and makes appropriate candidates of architecture or components.

Step 7, 8) For component files acquisition, component acquisition engine downloads those files from external component repositories and store them into the internal component repository.

The reason why it acquires ontology first is to increase performance of robots. If robots use physical files for measuring appropriation of components, it is overloaded to robots in that measuring time and resource costs are high. By measuring appropriation of components using component description level, performance of component search and acquisition can be improved [2].

4.6 Interaction within a Cell

As Fig. 6 shows, when the ontology repository for developers is updated, ontologies in cell manager of that group are also updated dynamically. After newly edited ontologies are classified by the cell manager automatically, cell manager notify that to human manager and he (she) decides weather it is correct, or modifies that classification by using UI of the repository manager tool. Cell manager provides initial common set of ontologies to developers, so that newly updated ontologies can be merged into cell ontologies easily.

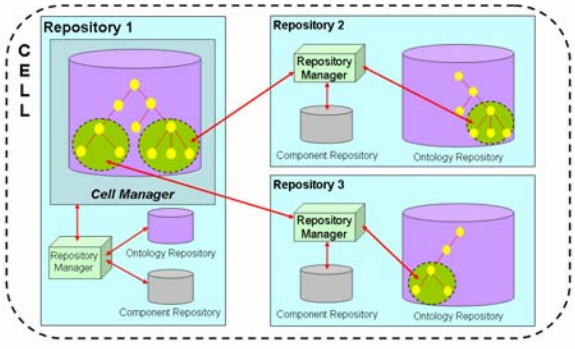

Fig. 6. **Interaction within a Cell**

4.8 Mapping of Ontology and Component Repository

As Fig. 7 shows, URIs (Uniform Resource Identifiers) of component instances in ontology repositories point at RDB of component repositories. In RDB of component repositories, component tables contain URIs, URLs, specifications, used count, and updated date. URLs point at locations of components within physical file system. Specifications point at documents and diagrams that are needed to understand components. Specifications consist of diagrams: UML diagrams, background Diagram (Conceptual), architecture (Inside of a component), and documents: interfaces (APIs), requirements specifications, design specifications, data for retirement (used count, updated date).

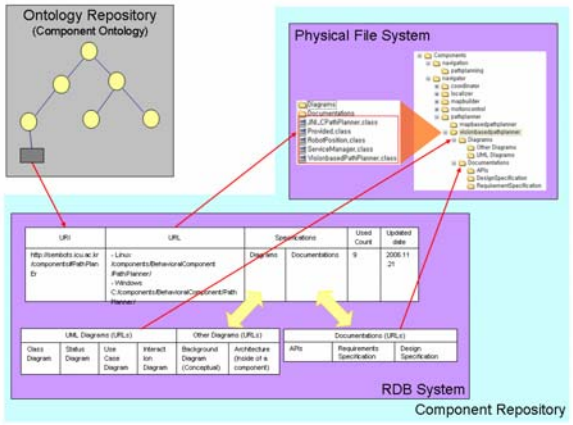

Fig. 7. **Connection of Ontology and Component Repository**

5. Implemented Prototype

Fig. 8 shows monitors for internal / external repositories. Left three elements of internal monitor show internal ontologies. Right monitor is for external repositories. Above part of the monitor shows component ontology and below part of the monitor shows component files those are stored in the external component repository. We will install these monitors into cells for showing acquisition process and evolution of the cell-based repositories. These monitors shows searching and acquisition process for Vision-based path planner.

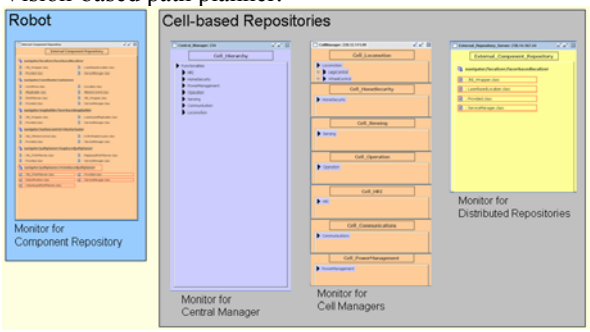

Fig. 8. **Internal/Cell-based external Repositories Monitors**

6. Conclusion and Future Work

A repository system is one of the essential elements to support self-growing robot software, in that it provides an opportunity for sharing components and experiences between robots, and it provides a management mechanism for component/application developers. In this paper, we described a cell-based approach for evolutionary component repositories for intelligent service robots. This cell-based approach was used to support evolutionary repositories. In this approach, we invented the concept of a cell that is a logical grouping of distributed repositories based on an ontology hierarchy. To accomplish evolutionary cells, we used a cell manager that integrated updated ontologies of distributed repositories, and branched cells dynamically. We also used a central manager to manage the hierarchy of cells. In addition, we described requirements, elements, an architecture, and a prototype for cell-based repositories.

Through this cell-based approach, we provide a framework of component repositories that can support self-growing robot software. A cell-based approach supports robots to keep necessary and essential components for robots in an evolutionary manner, and supports the evolution of functionalities from external distributed repositories for robots. A cell-based approach will improve the performance of searching and acquisition for developers and for self-growing action of robots. We will also achieve transparent sharing of components for robots and developers

We are currently researching the development of self-evolutionary cell-based repositories that can branch and regroup their cells automatically. We are also developing a web-portal based repository manager tool that supports registering, modifying and searching components for developers. In addition, we are developing an automatic classification mechanism for legacy components that are developed by other communities. Finally, we are doing an experiment for applying our cell-based repositories to the Infotainment Robot platform, then we will compare the cell-based approach with the previous experiment.

7. References

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

[4] 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.

[5] 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.

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

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

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

[9] 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.

[10] Regina M. M. Braga et al., "The Use of Mediation and Ontology Technologies for Software Component Information Retrieval", ACM, 2001.

[11] James X. Ci and Wei-Tek Tsai, "Distributed Component Hub for Reusable Software Components Management", Computer Software and Application Conference, COMSAC 2000, IEEE, 2000.

[12] 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.

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

21st International Conference on
Advanced Information Networking and Applications Workshops (AINAW'07)
0-7695-2847-3/07 $20.00 © 2007 IEEE