Software Modules Management Techniques for Multi-Cooperate Robots based on R-Object Model in Dynamic Environments

YunSik Son[1], YangSun Lee[2] and JinWoo Jung[1]*

[1]*Dept. of Computer Engineering, Dongguk University*
26 3-Ga Phil-Dong, Jung-Gu, Seoul 100-715, Korea

[2]*Dept. of Computer Engineering, Seokyeong University*
16-1 Jungneung-Dong, Sungbuk-Ku, Seoul 136-704, Korea

sonbug@dongguk.edu, yslee@skuniv.ac.kr,
**Corresponding Author: jwjung@dongguk.edu*

Abstract

In recent research, robots have been regarded as having a complete set of functions. Such robots can fulfill only specialized roles. Accordingly, it is not possible to use these robot systems for alternative purposes. The complexity of robot system for various tasks can be simplified by multi-robot cooperation system. Also In the classical robot motion paradigm, model-based paradigm in the field of motion planning of robots, robots make it difficult to respond efficiently to the dynamically variable environment such as disaster area. In order to handle such a situation that may be changed dynamically, a technology that allows a dynamic execution of data transmission and physical/logical connection between multiple robots based on scenarios is required.

In this paper, we deal with the software module management techniques for the multi-cooperate robot systems using R-Object model in dynamically changed environments. The proposed method is designed for managing versions of software modules to processing same or different functions and used to the multi-cooperate robot system can be adapt to any given environment and execute scenarios.

Keywords: *Robot Model, Reconfigurable Robot Software, Dynamic Environment, Robot Software Module Management*

1. Introduction

Industrial and alternative robots were the very important reason developing robots. Due to the reduced production costs of robots resulting from a constant development of robot technologies, robots that aid our everyday lives in various ways are becoming commercialized.

The various kinds of robots developed today employ the well-known methods of Sense-Plan-Act, Behavior-based, and Hybrid. These classical methodologies are difficult to deal with robots developed for disaster and space exploration purposes once their parts break down during operations, because traditional techniques is very complex to consider dynamically changing environments. Therefore, we would be able to respond more efficiently if we utilize surrounding robot components to assume the tasks that cannot be fulfilled by as single robot.

To solve such problems, many researches are conducted on systems that reconstruct robots through physical combination among modules. These systems construct robots through 2D or

3D physical linking based on a fixed control module [1, 2, 3, 4]. However, most module robot studies focus on only mechanical or physical conjunction and failed to propose a clear methodology as to how hardware-linked robots actually execute a given set of tasks.

An evolutionary robot technology is defined as each element as a robot component and allows it to accomplish its goals through a combination of these components. A method that effectively controls the dynamically shifting environment is needed be-cause such evolutionary robot executes a search of surrounding components, combination, separation, and task sequent reconstruction based on the given scenario [5].

An R-Object is modeling technique that represents an evolutionary robot that accommodates dynamic integration and separation while executing tasks.

In this paper, we will discuss on robot software management techniques in dynamically changing environment. The robot software management techniques are based on R-Object model and solve the problem for software management in dynamically reconstructable robots.

2. Evolutionary Robots using Multi-Robot Cooperation

Evolutionary robots have a self-developing mechanism that allows them to adapt to different environments through learning and evolution. In the notion of evolutionary robots, we can regard each functional unit as an independent component robot. Functional unit robots execute a given set of tasks through mutual collaboration with surrounding robots [6, 7, 8, 9]. A methodology for accomplishing given goals is required in order to develop an evolutionary robot system. In this study, we limit robots to those used in natural disasters and space explorations.

Table 1. Scenario Execution Steps in Evolution [10]

Step	Description
Step 1	A scenario is given.
Step 2	Separate the scenario into task sequences.
Step 3	Each robot diagnoses whether it can accomplish a given t ask. At this time, robots use task sequences and task-beha vior mapping information.
Step 4	Decide major control robot in the given robot set.
Step 5	Robots execute a scenario. In this process, robots needed for a task cooperate with each other.
Step 6	If a cooperating robot is removed or a new robot is added, the system reconstructs the robot set and goes to step 4 u ntil the scenario is ended.
Step 7.	If, during scenario execution, other robots are equipped w ith newer or enhanced functions and if a more effective sc enario execution seems possible based on the given metri cs, the particular robot reconstructs the task sequence for the current scenario execution and the structure of the rob ot function layer.

Table 1 shows the scenario execution procedure chosen based on the analysis of space exploration robots [11].

The layer of the proposed evolutionary robot can be divided in a robot platform-dependent layer and a platform-independent layer, which gives the advantage of being able to deal with various platform-dependent functions using the same commands in the platform-independent layer. We designed a new robot software module layer model called TBPPC (Task-Behavior-

PIF-PSF-Component) in previous work [10, 12, 13, 14] for evolutionary robots. Figure 1 illustrates the layers of a robot analyzed from a functional perspective and it shows the process of modifying the mapping relationship when an error occurs during mapping actual robot components run based on the TBPPC model.

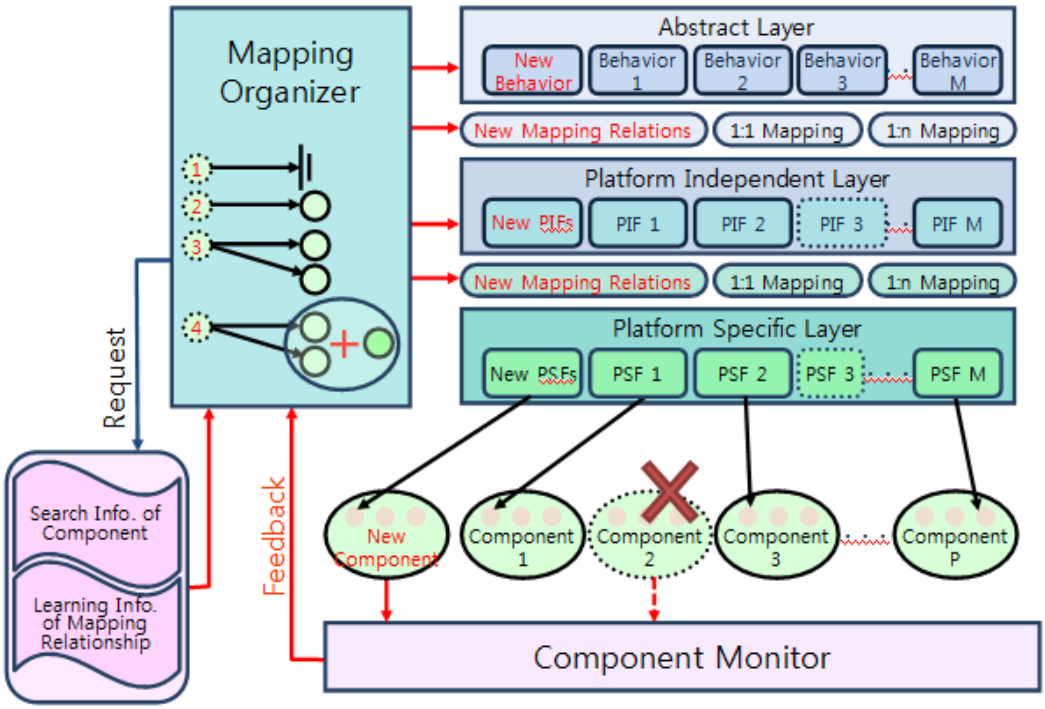


Figure 1. Functional Perspective Robot Layer

3. R-Object Model

We need a new robot model because it is difficult to reconstruct robot components for scenario execution with the existing robot model. In this study, we define the unit with functions including communication, self-diagnosis, and inference as a single robot model and we propose an R-Object model for component-based evolutionary robots [10, 12, 13, 14].

The R-Object model is consisted of a task-diagnostic module that determines if a task can be executed, TBPPC mapping information, an action algorithm, and attribute information with which robots are expressed. The task-diagnostic module recognizes the functionality of the robot self and identifies the functions needed additionally to perform the given tasks. TBPPC mapping information shows the mapping relationships across tasks, behaviors, PIFs, PSFs, and components. Further, PIF (Platform Independent Function) and PSF (Platform Specific Function) each relate to expressing hardware-independent functions and hardware-dependent functions, respectively. Also, individual robot components can contain multiple PSFs. Each robot model is designed to enable independent behavior and collaboration through restructuring.

The R-Object is modeled while taking into consideration the defined input data and hardware robot system. Figure 2 illustrates the R-Object model.

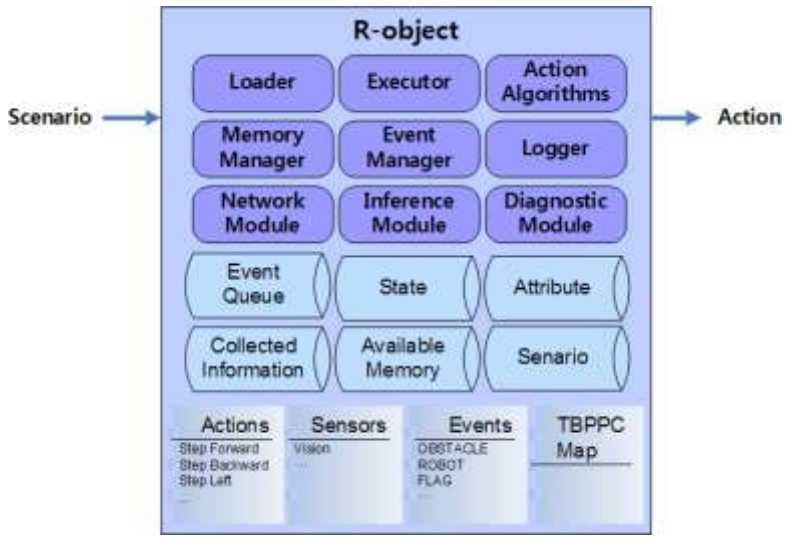


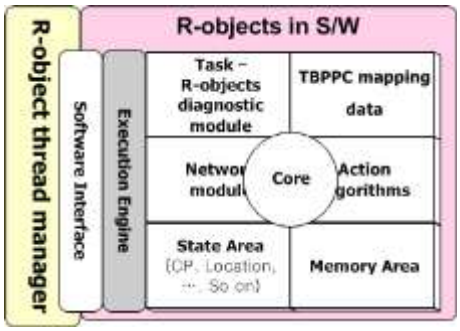
Figure 2. R-Object Model

The R-Object is consisted of 6 memory sectors, 4 tables, and 9 modules. A memory sector expresses the state and the attribute information of a robot and is consisted of an event queue for executing scenarios based on events and a memory sector for saving information acquired during the scenario execution. The 4 tables are composed of actions, mounted sensors, possible events, and mapping information across task-components.

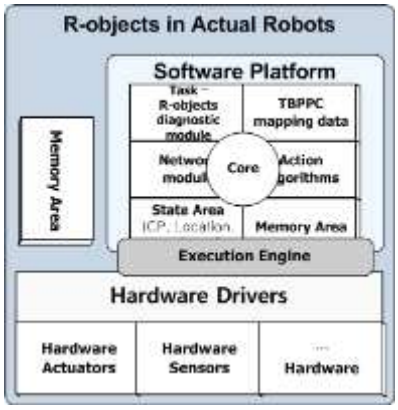
The 9 modules that carry out the core tasks in the R-Object model include the loader, executor, action algorithm, memory manager, event manager, network module, reasoning module, diagnosis module, and logger. The loader delivers data to the executor, which analyzes scenarios received by the robot. The action algorithm role is to contain a set of actions that can be taken by the robot. The memory manager is a module that manages memory loaded on to the robot and the event manager is a module for capturing and filtering events generated from the robot. The network module is responsible for inter-robot communication and the reasoning module consists of a set of functions that can be replaced with when the robot breaks down. The diagnosis module determines what tasks the robot object can execute based on the mapping information and sub-task sequences, which result from scenario analysis. Once the diagnosis module decides that a particular robot is able to carry out a task, it creates a set of robots necessary for the task. The logger denoted by the dotted line is a module that collects debugging information and it can be added as needed.

With the R-Object model, users can generate specific modeling information for an actual robot. The R-Object model offers a way to generate new models by extending the existing model while taking reusability into consideration. Also, the R-Object model can be extended through connection of R-Objects during scenario execution.

Input data for R-Object modeling consists of name, status, attribute, action, sensor, and network information. The status information of a robot includes information subject to change during scenario execution such as control point, location, and declination. The attribute information of a robot is defined as density, material, color, width, height, and shape. Action and sensor information indicates the number and types of actions and sensors that a robot can perform.



(a) On Software



(b) On Hardware

Figure 3. R-Object Model on Software and Hardware

The goal of the model is to abstractly portray a hardware robot and to describe it with consistency from a functional perspective independent from its hardware. It also offers a TBPPC model-based hardware-independent interface so that dynamic aggregation and separation of software and hardware is possible. This allows us to apply identical models to software and hardware. Figure 3 illustrates the structure with which the R-Object modules are mounted on software and hardware.

In order to link and separate robot components, we must consider the information linking structure between R-Objects. In this structure, we design the linking structures of configuration information conservation form, tree form, and star form and we analyze the advantages and disadvantages of each. For the analysis, we assume that robots are interlinked with one another in a linear form as shown in (a) of Figure 4.

The method of the configuration information conservation form (a) is expressed identically as the link structure of actual robots. Thus, it is an advantage that we can maintain the configuration information for robot linking. However, there exists overhead from the communication between the intermediary robots (B, C) and the major control robot (A*) to direct orders to an indirectly linked robot (D). In a tree structure (b) link, the major control robot can directly place orders to all linked robots but non-major control robots cannot put direct orders to other robots and information about the type of robot link becomes lost. In the star-type link (c) structure, robots can send directions to one another and they can directly and mutually exchange information. However, the star-type structure fails to maintain the information about the type of robot link.

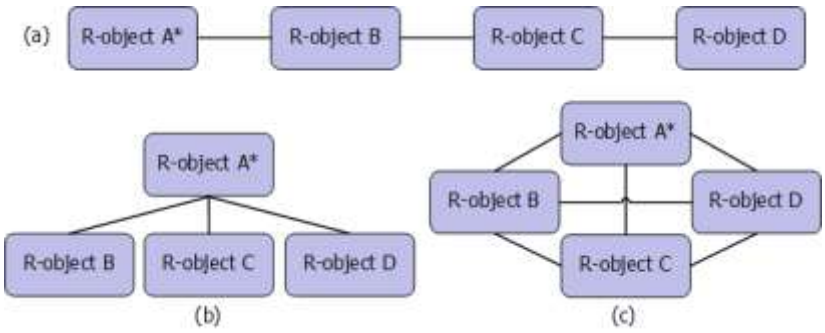


Figure 4. Representation of Connection Structure

In a component-based evolutionary robot, the robot in control sends orders to other linked robots and assigns them tasks. The configuration information conserving structure can always maintain configuration information but it is not suitable because it entails overhead for ordering commands. Also, the star-type is not appropriate for the notion of component-based evolutionary robots because it allows all robots to send orders to one another on top of the spatial overhead from expressing so much connection information. Thus, we chose the tree-type structure, which well reflects the idea of linking and separation of component-based evolutionary robots and conserved the linking information of each robot in order to maintain our configuration information.

Information sharing across individual robots is done by communication through network modules. Also, all information can be shared since linked robots are logically expressed by a single R-Object.

The robot used in this case study was a mobile robot platform with the R-Object model as shown in Figure 5.

Figure 5. Experimental Robot Platform

4. Robot Software Module Management Techniques

In this chapter, we propose the management techniques for robot software modules based on R-Object model. Especially, in this study, we are focusing on the management method for software unit modules that executing same function by robot itself in dynamically changing environments. In the proposed method, all robots based on R-Object has a software modules information graph to process a given specific task and expand robot group's functions by merging graphs of each joined robot for cooperation work.

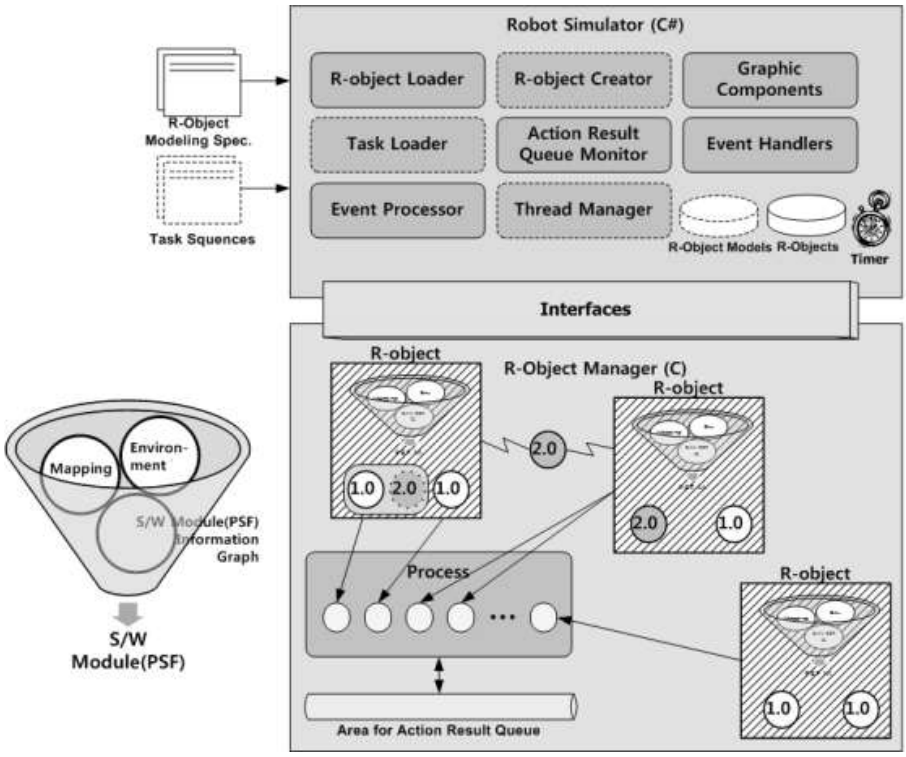


Figure 6. Extended R-Object Simulation Model

Next, we deal the software module graph and operations for graph used in proposed method. Table 2 shows notations in graph management operations. In the proposed method, software modules information graph, G is directed acyclic graph.

Table 2. Notations for Software Modules Information Graph

Symbol	Description	Symbol	Description
G	G = { (v, E)* }	Eo	Set of outer edges
V	Set of vertex(node)	I	Module information
v	Vertex(node), v = (I, T, M)	T	Type of function (self, cooperation, other)
E	Set of edges	M	Meta information
Ei	Set of inner edges	S	Super vertex for all vertex, no inner edges

The Algorithm in Table 3 are to generate module selection information.

Table 3. Module Selection Information Generation Algorithm

Input: G
Output: order information of G

1. Write lower vertexes while traverse graph G from bottom to top.
2. Except an S vertex.

The algorithm to merge information graphs is follows. We can apply this algorithm to both single robot module information graph and merged graph.

Table 4. Information Graphs Merge Algorithm

Input: $G_1$and G_2
Output: merged graph G_m

1. Calculate module information of $G_1$and G_2.
2. Add all vertexes v of G_1 and G_2 to new graph G_m except common vertexes of G_1 and G_2.
3. Add E_i and E_o of common vertexes of G_1 and G_2 to graph G_m
4. Find vertexes that have no Ei from $G_1$and G_2. (find root nodes)
5. Check cross element - $v_1 \notin G_1$ and $v_2 \notin G_2$ – for found vertexes at step 4.
6. If step 5 is true than add v_1 and v_2 to Eo of new vertex S and add S to Ei of v_1 and Ei of v_2.
   a. If $v_1 \notin G_2$ and $v_2 \notin G_1$ then
      E_i of $v_1 \cup S$ and E_i of $v_2 \cup S$
      E_o of $S \cup v_1$ and E_o of $S \cup v_2$
      $\{ v_1 \mid v_1 \in G_1$ and E_i of $v_1 = \Phi \}$
      $\{ v_2 \mid v_2 \in G_2$ and E_i of $v_2 = \Phi \}$
   b. If S is already existed then E_o of new $S \cup E_o$ of old S and delete old S.
7. If the graph has cycles then process normalization to remove the cycles.
      criteria1. Remove E_o of the most deepest v in the cycle.
      criteria2. Remove E_o of v has the most smallest sub v in the cycle.
      criteria3. Reflect the depth of original graph.
8. Calculates the module selection information of the new constructed graph.

By using generated information and graph, the multi cooperative robot system can select a suitable robot software unit module on the situation.

Figure 7 shows result that merge graph from graph G1 and G2 using algorithm in Table 4. In Figure 7, each graphs are represented by set of tuples as the definition in Table 2 and each tuples are consist of (vertex, (set of inner edge, set of outer edge)).

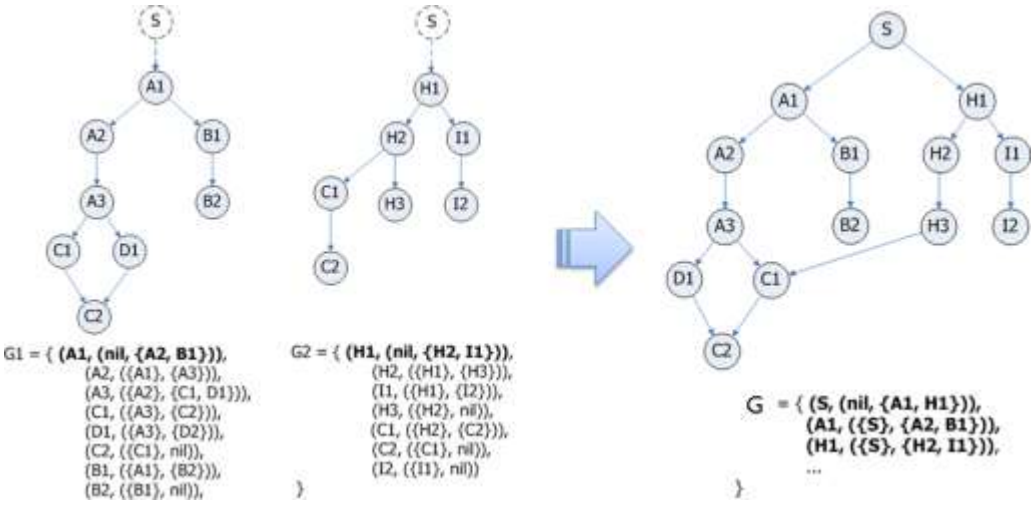


Figure 7. Merge Operation on Software Module Information Graph

Table 5. Module Selection Information Generation Algorithm

Input: G Output: order information for all vertex of G 1. write all lower vertex information of visited one while traverse from bottom vertexes to the top vertex. 2. excepts S vertex.

Table 6. Candidate Module Information Extraction Algorithm

Input: module version information, module selection information (if merged graph, includes module selection information of pre-merged graph). Output: candidate module information 1. Union order information of the each module by inputted module version information, module selection information. 2. Union order information until not changed version information (information are increased while repeated, and cycled vertex information are generated finally).

Next, the module selection information is generated from the software module information graph and the candidate module information is extracted from it. Table 5 shows the algorithm to generate module selection information from the graph and Table 6 is the algorithm to extract candidate information.

Figure 8 shows the module selection information generated from each graphs using algorithm in Table 5.

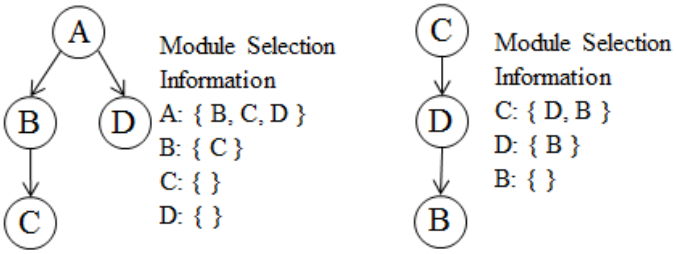


Figure 8. Simple Software Module Information Graphs and Module Selection Information

Next example is about to introduce graph normalization of merged graph. Figure 8 is target software information graph and Figure 9 is merged graph of Figure 8 and resolved cycle problem by proposed method. The directed edge from node D to node B is eliminated according to the criteria 1.

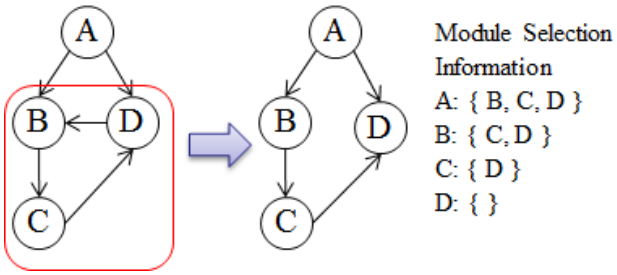


Figure 9. Cycle Resolution for Merged Software Module Information Graph

Information obtained through this process, a software module graph is utilized in the scenario execution of the robots. And, each candidate functions are evaluated by features - environmental information, execution complexity, performance and etc. – and the robot can choose the more efficient function to execute target sub task sequence.

5. Conclusions

In the classical robot paradigm, robots make it difficult to apply efficiently to the dynamically variable environment such as disaster environment. An R-Object model could adapt to changing environments and features dynamic linking, separation, and collaboration. In this study, we proposed robot software management techniques in dynamically changed environments. Proposed techniques can solve various software problems while robot reconstruction to process the given task.

In the future, we need to study how to determine optimized function for given task and situation. Also, we will need to simulate and experiments about the R-Object model and suggested techniques.

Acknowledgements

This paper was extended from the previous research paper “Robot Software Modules Management Techniques in Dynamic Environments” in AITS-MSA 2012.

References

[1] A. L. Christensen, R. O'Grady and M. Dorigo, "Distributed Growth of Specific Structures Using Directional Self-Assembly", IEEE Robotics & Automation Magazine, **(2007)**, pp. 18-25.
[2] C. Detweiler, M. Vona, Y. R. Yoon, S. K. Yun and D. Rus, "Self-Assembling Mobile Linkages", IEEE Robotics & Automation Magazine, **(2007)**, pp. 45-55.
[3] S. Murata, K. Kakomura and H. Kurokawa, "Toward a Scalable Modular Robotic System", IEEE Robotics & Automation Magazine, **(2007)**, pp. 56-63.
[4] J. H. Kim, M. T. Choi, M. S. Kim, S. T. Kim, M. S. Kim, S. Y. Park, J. H. Lee and B. K. Kim, "Intelligent Robot Software Architecture", IEEE International Conference on Robotics and Automation, **(2007)**, pp. 385-397.
[5] T. A. Choi, "Scenario based Robot Programming", IASTED International Confer-ence on Robotics and Applications, **(2006)**, pp. 55-60.
[6] L. E. Parker, "Adaptive Heterogeneous Muti-Robot Teams", Neurocomputing, Special issue of NEURAP '98, Neural Network and Their Applications, vol. 28, **(1999)**, pp. 75-92.
[7] D. Kim , S. Park , Y. Jin , H. Chang , Y. S. Park , I. Y. Ko , K. Lee , J. Lee , Y. C. Park and S. Lee, "SHAGE: a framework for self-managed robot software", Proceedings of the 2006 International Workshop on Self-Adaptation and Self-Managing Systems, **(2006)**.
[8] D. Kim and S. Park, "Alchemistj: A Framework for Self-Adaptive Software", The 2005 IFIP International Conference on Embedded And Ubiquitous Computing, LNCS3824, **(2005)**, pp. 98-109.
[9] M. Karanam and A. R. Akepogu, "A Framework for Visualizing Model-Driven Software Evolution – Its Evaluation", International Journal of Software Engineering and Its Applications, vol. 5, no. 2, **(2011)** April, pp. 136-148.
[10] J. W. Park, Y. S. Son, J. W. Jung and S. M. Oh, "Software Interface for Hardware-independent Robot Platforms", International Journal of Assistive Robotics and Mechatronics, vol. 9, no. 4, **(2008)**, pp. 110-119.
[11] M. Romano, B. N. Agrawal and F. Bernelli-Zazzera, "Experiments on command shaping control of a manipulator with flexible links", Journal of Guidance Control and Dynamics, vol. 25, no. 2, **(2002)**, pp. 232-239.
[12] Y. S. Son, J. W. Park, J. W. Jung, and S. M. Oh, "R-Object Model Simulator for Evolutionary Robots", The 6th International Conference on Ubiquitous Robots and Ambient Intelligence, URAI 2009, **(2009)**.
[13] J. W. Park, Y. S. Son, J. W. Jung and S. M. Oh, "R-Object Model for Evolutionary Robots using Multi-robot Cooperation," The 2nd IFAC International Conference on Intelligent Control Systems and Signal Processing, **(2009)**.
[14] J. W. Park, Y. S. Son, J. W. Jung and S. M. Oh, "A Study on the Evolution of Software Aspects for Evolutionary Robots", International Journal of Assistive Robotics and Systems, vol. 10, no. 2, **(2009)**, pp. 27-35.
[15] N. Koeniq and A. Howard, "Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator", Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 3, **(2004)**, pp. 2149-2154.
[16] O. Reinoso, A. Gil, L. Paya and M. Julia, "Mechanisms for collaborative teleoperation with a team of cooperative robots", Industrial Robot, vol. 35, no. 1, **(2008)**, pp. 27-36.
[17] J. W. Jung and B. C. So, "An Idea to Reduce Number of Cells in the 2D Exact Cell Decomposition-based Mobile Robot Path Planning", The 7th International Conference on Ubiquitous Robots and Ambient Intelligence, **(2010)**.
[18] J. H. Lim, S. H. Song, J. R. Son, H. S. Park and H. S. Kim, "An Automated Test Method for Robot Platform and Its Components", International Journal of Software Engineering and Its Applications, vol. 4, no. 3, **(2010)**, pp. 9-18.
[19] N. Azoui and L. Saidi, "Passivity Based Adaptive Control of Robotic Manipulators Electrically Controlled", International Journal of Advanced Science and Technology, vol. 34, **(2011)**, pp. 45-54.
[20] S. M. Ghosh, H. R. Sharma and V. Mohabay, "A Study of Software Change Management Problem", International Journal of Database Theory and Application, vol. 4, no. 3, **(2011)**, pp. 39-48.

Authors

Yunsik Son

He received the B.S. degree from the Dept. of Computer Science, Dongguk University, Seoul, Korea, in 2004, and M.S. and Ph.D. degrees from the Dept. of Computer Engineering, Dongguk University, Seoul, Korea in 2006 and 2009, respectively. Currently, he is a Researcher of the Dept. of Computer Science and Engineering, Dongguk University, Seoul, Korea. His research areas include smart system solutions, secure software, programming languages, compiler construction, and mobile/embedded systems.

YangSun Lee

He received the B.S. degree from the Dept. of Computer Science, Dongguk University, Seoul, Korea, in 1985, and M.S. and Ph.D. degrees from Dept. of Computer Engineering, Dongguk University, Seoul, Korea in 1987 and 2003, respectively. He was a Manager of the Computer Center, Seokyeong University from 1996-2000, a Director of Korea Multimedia Society from 2004, a General Director of Korea Multimedia Society from 2005-2006 and a Vice President of Korea Multimedia Society in 2009. Also, he was a Director of Korea Information Processing Society from 2006, and a President of a Society for the Study of Game at Korea Information Processing Society from 2006. And, he was a Director of Smart Developer Association from 2011-2012. Currently, he is a Professor of Dept. of Computer Engineering, Seokyeong University, Seoul, Korea. His research areas include smart system solutions, programming languages, and embedded systems.

Jin-Woo Jung

He received the B.S. degree from the Dept. of Electricity and Electronic Engineering,Korea Advanced Institute of Science and Technology, Daejeon, Korea, in 1997, and M.S. and Ph.D. degrees from the Electricity and Electronic Engineering, , Korea Advanced Institute of Science and Technology, Daejeon, Korea in 1999 and 2004, respectively. Currently, he is a Professor of the Dept. of Computer Science and Engineering, Dongguk University, Seoul, Korea. His research areas include human-robot interaction, assistive robotics, and embedded systems.