Proposed environment to support development and experiment in RoboCupRescue Simulation

Shunki Takami[1], Kazuo Takayanagi[1], Shivashish Jaishy[1], Nobuhiro Ito[2], Kazunori Iwata[3], Yohsuke Murase[4], and Takeshi Uchitane[5]

[1] Graduate School of Business Administration and Computer Science, Aichi Institute of Technology, Japan
[2] Department of Information Science, Aichi Institute of Technology, Japan
[3] Department of Business Administration, Aichi University, Japan
[4] RIKEN Advanced Institute for Computational Science, Japan
[5] Research Institute for Economics and Business Administration, Kobe University, Japan

Abstract. The RoboCupRescue Simulation project is a test bed for multi-agent systems research for disaster relief. However, researchers have to implement many types of algorithm and require a complicated procedure for experiments, which places a heavy burden on them. Therefore, we propose an environment that integrates an agent-development framework and an experiment-management system to support researchers.

Keywords: Programming environment, Experiment management, Rescue simulation

1 Introduction

In recent years, the annual RoboCup international robotics competition has hosted the RoboCupRescue Simulation (RRS) project to confront large-scale natural disasters [1]. In particular, the Agent Competition is a platform for studying disaster-rescue agents and simulations. Our aim is to contribute to society by submitting results for this project.

However, in order to solve the disaster-relief problem targeted by the RRS, it is necessary to implement a combination of multiple algorithms such as those for path planning, information sharing, and resource allocation. In the experiment, numerous simulations are required for multiple disaster areas and changes in the various conditions of the area, such as fire, building collapse, and communication. Because these are burdensome for researchers, we propose an environment that integrates an agent-development framework and an experiment-management system. In the evaluation, we confirm that code re-usability and researcher burden were reduced in the experiment.

2 Research and development in RRS

2.1 Overview of RRS

The RRS is a research platform that simulates disaster situations and disaster-relief activities on a computer. It can handle disaster-relief activities over roughly five hours from the occurrence of a disaster.

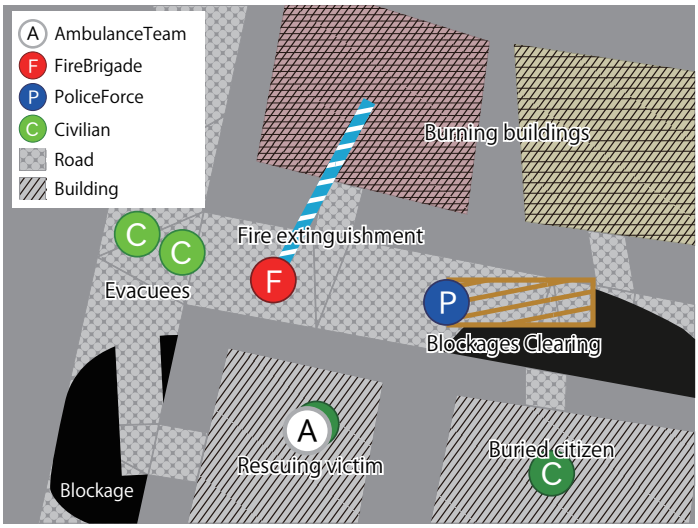

Fig. 1. Overview of RRS

Figure 1 shows the activities of agents in the RRS. In the disaster-relief activities, we control six types of agents, namely AmbulanceTeam, FireBrigade, PoliceForce, and the headquarters of each of these units. In addition, there are agents to simulate disaster situations, namely Civilian agent.

- **AmbulanceTeam and AmbulanceCentre**
  These agents rescue other agents that cannot move by themselves.
- **FireBrigade and FireStation**
  These agents extinguish fires in buildings.
- **PoliceForce and PoliceOffice**
  These agents clear road blockages.
- **Civilian**
  In the competition, this agent moves automatically to evacuation centers.

By using the RRS, it is possible to research applications of artificial intelligence and information science to natural-disaster rescue problems. Researchers have been investigating algorithms for route searching, information sharing, and task allocation in a disaster situation. In the RRS project, five tasks are advocated especially, namely Group Formation, Path Planning, Search, Multi-Task Allocation, and Communication. Every year, competitions using agent programs are held for the purpose of technical exchange.

2.2 Agent development in RRS

The disaster-relief problem handled by the RRS is a complex problem as the damage situations, such as fire, building collapse, and the availability of wireless communication, change from moment to moment in afflicted areas. These

changes are addressed by the disaster-relief strategies of teams of disaster-relief robotss that differ according to the disaster situation. To construct a disaster-relief strategy, it is necessary to prepare all the algorithms for tasks such as route searching, information sharing, and resource allocation in the disaster environment. Moreover, in activities such as the PoliceForce clearing blockages, it is necessary to use angles and coordinates to specify the direction for activity and the positions of the agents.

To promote research involving the RRS, it is necessary to clarify the structure of a complicated disaster-relief problem and subdivide it before solving it.

2.3 Experiments in RRS

To develop and evaluate RRS agents, it is necessary to conduct experiments on multiple disaster areas while considering various conditions such as the locations of fires, the rate at which buildings collapse, and communication situations. Also, the parameters of an agent may be set for each of these situations. All this requires numerous simulations.

In the RRS, each simulation takes around 20 min to execute. Thus, multiple computing hosts may be simulated simultaneously to improve the efficiency of the experiment. It is common to distribute the agent process to more computers in one simulation, so it is necessary to control all these computers simultaneously. However, since the time synchronization between the simulator components is based on the command transmission of the kernel, there is no need to externally control it. Therefore, we externally control just the activation and termination of components.

2.4 Related work

OpenRTM-aist OpenRTM-aist[2] is a framework for robot development that was developed by the National Institute of Advanced Industrial Science and Technology and whose implementation is based on the RT-Middleware standard [3]. This common platform standard divides robot elements such as actuator control, sensor input, and algorithms necessary for behavior control into single components (This is called RT-Component:RTC), and then constructs a robot by combining all such components. This makes it possible to subdivide the elements that are necessary for controlling the robot. Because each component can be exchanged as a module and existing modules can be included, it is possible to reduce the burden on developers in the development and improvement of robots.

Because RT-Middleware is applied mainly to real robots, it is suitable for developing robots that are controlled in real time. It is difficult to utilize existing code and knowledge when adopting RT-Middleware because RRS agents are programed mainly with a sequential structure.

Hinemos Hinemos is software that was developed by NTT Data Corporation to monitor and manage the job execution of multiple computing hosts [4]. We can automate the execution of an experiment by treating the experiment as each process, and it is also possible to manage the execution on multiple calculation hosts. Hinemos allows the management of issues such as job exit codes, job stylization, advanced experiment scheduling, and GUI execution management. However, because Hinemos is high-function general-purpose software used for various computer-based business tasks, it has a high operational cost if used for RRS experiments.

OACIS OACIS is a simulation-execution management framework developed by the discrete-event-simulation research team of the RIKEN Advanced Institute for Computational Science [5]. This software manages jobs in a similar way to Hinemos. In particular, it has a job-management function that specializes in the simulation and management of experiment results, and a function that specializes in the execution of simulations. It is possible to use OACIS to support numerous simulations and to perform analysis with various conditions by managing the experimental parameters and automatically managing the results.

However, because OACIS is a general-purpose system for various types of simulation software, complicated operations are required to execute RRS simulations. The creation of simulation scripts, the agent programs, and the map and scenario files must be managed by other methods, making OACIS complicated to use for the RRS.

3 Proposal and implementation

3.1 Research objective

In this paper, which is based on the current state of RRS agent development and experiments, we implement an agent development framework by introducing a modular structure to clarify and solve the complicated problems associated with disaster relief. We also propose the framework as our community standard, which makes it easy to reuse program code. Furthermore, to reduce the experimental burden, we implement an RRS experimental environment that is based on the OACIS simulation-execution management framework. We then combine the agent framework and the RRS experimental environment. Finally, we propose an environment that offers comprehensive support for agent development and experiments.

3.2 Proposal of environment to support development

Design of agent-development framework In order for many researchers to clarify and solve complicated problems, we modularize part of the program code. To make it easier to reuse the program code and to reduce the burden on researchers, an agent-development framework is desired. A framework-design

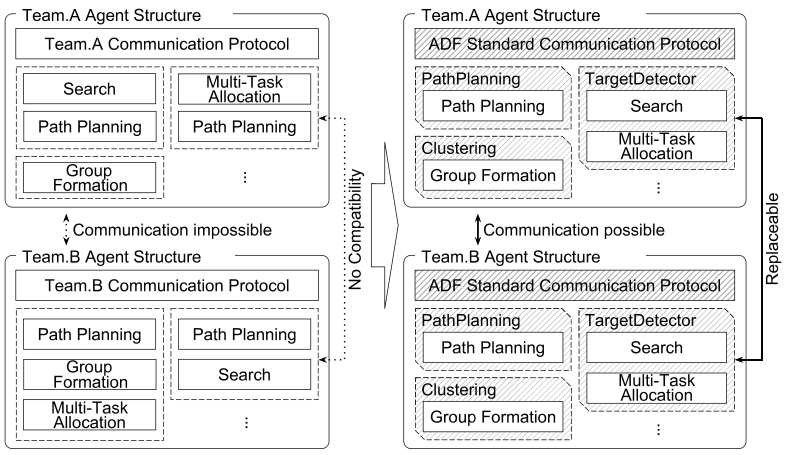

Fig. 2. Before and after introducing common architecture

method based on OpenRTM-aist (mentioned in Sect. 2.4) would also be conceivable. However, RRS agents are programmed in a sequential structure, so we propose and design a unique Agent Development Framework (ADF) that makes it easier to utilize existing program code and knowledge.

Introduction of a common agent architecture By defining the overall behavior of an agent as a common architecture, we reduce the differences in combinations of components by each developer and ensure re-usability. This allows developers to implement modules based on this common architecture when developing agent programs.

Figure 2 shows the architecture before and after introducing this common architecture. The left-hand portion of the figure shows the existing agent structure, whereas the right-hand portion introduces the common architecture. The portability of the existing program is low because each researcher builds an agent program independently according to individual research agendas. We have commonized the agent-program structure, which is the shaded part of the figure, so that the program code can be re-used easily. Also, this unifies the communication protocol between agents and enables communication with agents developed by others.

Modularization of program code As with the RT-Middleware discussed in Sect. 2.4, component modularization is introduced to reduce the burden on researchers and to make it possible to reuse the program code used in agent development.

– **Algorithm modularization**

At the present stage of modularization, we divide as much as possible based on the five tasks presented in the RRS project. Figure 3 shows the

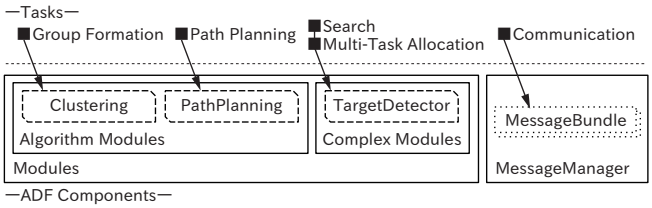

Fig. 3. Relationship between RRS tasks and ADF components

relationship between RRS tasks and ADF components. The top portion of the figure contains five tasks, and the bottom portion contains the framework components. We classify algorithms for solving complex problems and algorithms for solving simple problems as Complex Modules and Algorithm Modules, respectively, thereby clarifying the directionality of each module. We view algorithms to solve complex problems (Complex Modules) as aggregates of simple problems. Thus, the modules should be programmed by dividing the structure inside the program code as much as possible. Moreover, if it is found empirically that the program can be divided in the future, we aim to clarify complicated problems as new modules.

– **Modularization of control program**
  As described in Sect. 2.2, it is necessary to control an agent by specifying such properties as its coordinates and angles. Low-level agent control is modularized as a control program. By separating macro algorithms such as decision making and micro algorithms such as control using the coordinates and angles, we reduce the burden on researchers who wish to study a single algorithm such as the decision-making one.

Other approaches By introducing a common architecture for agents, it becomes possible to manage the inter-agent communication protocol and various agent parameters collectively.

– **Unifying inter-agent communication protocols**
  In the RRS, the inter-agent communication protocols are currently not unified, which strengthens the dependency between components makes it difficult to modularize the algorithm. Therefore, with reference to the RRS inter-agent communication protocols proposed by Ota et al. [6] and Obashi et al. [7], we define a common inter-agent communication protocol. We define messages communicated under this protocol as members of either an information-sharing family or a command family. In the information-sharing family, information about agents, roads, and buildings is shared. In the command family, commands for relief, fire extinguishing, blockage clearing, and searching are ordered.

– **Collective management of parameters**
  In the framework, we commonize the parameter-input interface by collectively managing the designation of modules and parameters in the algorithm

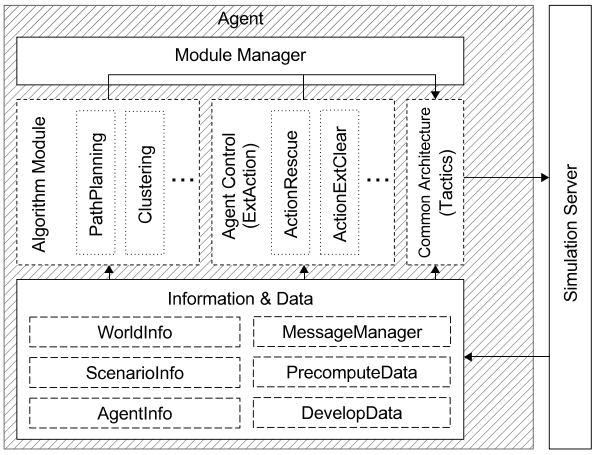

Fig. 4. Structure of framework

that are to be changed at the time of the experiment. This makes it easy to input parameters from tools that support the experiments, such as OACIS described in Sect. 2.4.

3.3 Implementation of agent-development framework

Implementation overview Figure 4 shows the structure of the agent-development framework that is implemented. The internal aspects of the agent, as represented by the shaded part in the figure, are implementations of the framework. The parts indicated by dotted lines are modules, which are the objects to be programmed at the time of agent development. In this section, we describe each implementation of the framework based on the design discussed in Sect. 3.2.

Common agent architecture This component defines how agents invoke algorithm modules and control-program modules to determine behavior. This component is referred to as Tactics. Each agent determines its behavior by invoking elements such as task assignment, a route-search algorithm, and a control-program module within Tactics. In addition, each module is invoked by its alias name and can select the module to be attached by the agent-configuration file.

Modules In Sect. 3.2 shows that each task is modularized as Clustering, PathPlanning, or TargetDetector. Also, the control program is modularized as ExtAction. An instance of the module is created and managed in a component called the ModuleManager. This further enables switching the module to be used by loading the module-configuration file (module.cfg) at agent startup.

Collective management of parameters The settings of the module to be loaded are obtained from the aforementioned file module.cfg. In addition, parameters in algorithms can be obtained from the DevelopData component, and

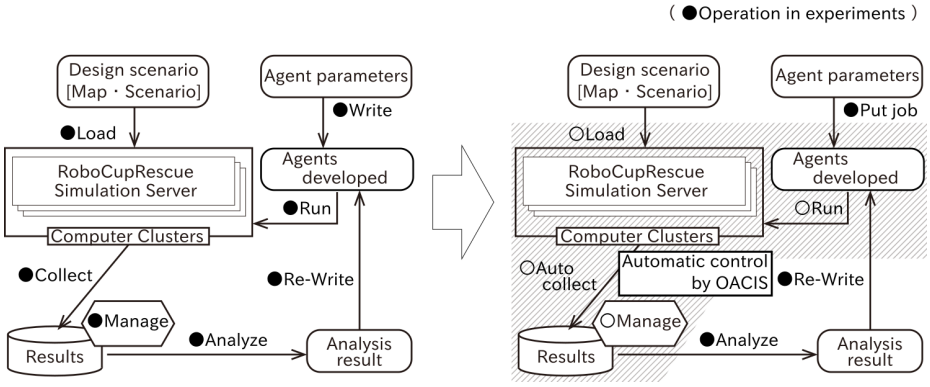

Fig. 5. Before and after introducing experimental management by OACIS

each value can be input by JSON [8] formatted text as an argument at agent startup.

3.4 Proposal of environment to support experiments

To realize the RRS experimental-support environment, we introduce experiment management using the OACIS simulation-execution management framework.

In Fig. 5, the left-hand portion of the figure shows the operations before the introduction of OACIS experimental management, whereas the right-hand portion shows the operations after its introduction. To experiment with an agent in the RRS, it was necessary to designate the parameters of various agents, control computer clusters to execute simulation, and collect and manage numerous results as indicated by the black points. By introducing experimental management by OACIS, the work shown in the shaded area in the figure is automated and the operation performed in experiments can be reduced. Moreover, by automating the experimental operation, it becomes easier to reproduce the experiment.

However, because OACIS is a general-purpose system, there are insufficient functions that can be applied to RRS experiments. Therefore, we develop functions to be supplemented when applying OACIS to RRS experiments as an extension to the RRS. This section describes these supplemental functions. This solves the complicated problem that OACIS to use for RRS described in Sect. 2.4.

- **Agent management**
  Although OACIS can manage simulators and experimental parameters, it cannot manage the agent program itself. Therefore, we develop an extension to manage the agent program.
- **Map and Scenario management**
  OACIS cannot manage maps and disaster scenarios as can agent programs. Therefore, we develop an extension to manage maps and disaster scenarios.
- **Computer cluster management**
  In the RRS, one simulation is executed in a computer cluster, combining many computers. Although OAICS supports job execution in computer

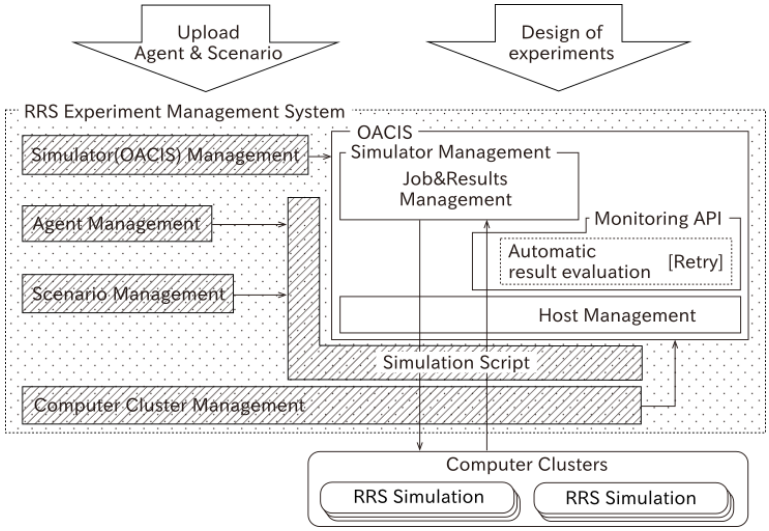

Fig. 6. Extensions of OACIS for RRS

clusters, it does not conform to the protocol of launching processes required by the RRS program. We developed a mechanism to adjust an interface to launch the RRS program using OACIS.

– **Simulation script**
  It is necessary to prepare a script that describes a series of actions in a simulation, which can be executed from OACIS. This script loads an agent and MapScenario, connects to each computer of the computer cluster, and executes the simulation. This enables it to treat experiments as a single job and enables management by OACIS.

– **Simulator management**
  In OACIS, a simulation script and a set of parameters that can be specified are known collectively as a Simulator. Parameters change depending on the purpose of the experiment, such as the agents to be used, the modules to be used, and the parameters of the algorithms. Therefore, we develop a mechanism to register Simulator with OACIS according to various experimental methods.

3.5 Extensions of OACIS for RRS

In implementing the extension to sustain maintainability, we did not modify the OACIS body program, which is controlled by the extension part through an API provided by OACIS. Figure 6 shows a conceptual image of the OACIS extensions. The shaded areas in the figure are extensions to the RRS.

– **Agent management**
  This extension unit enable to upload the agent program body in the directory under management. Then, we register and manage the agent's name

and storage location, parameters, comments, and registration date and time in the database of the extension unit. This enables to manage the agent files in an integrated way.

– **Map and Scenario management**
  This extension unit enable to upload the map and scenarios in the directory under management, similarly as with the agents. Then, we register and manage the name and storage location, comments, and registration date and time in the database of the extension unit.

– **Computer cluster management**
  Information about the computer cluster is managed in the database of the extension unit. When executing the simulation, the simulation script is invoked on the local host and simulation is executed using each computer. The simulation script connects to each designated computer according to the configuration file of the computer cluster that is arranged in each working directory. Therefore, switching of computer clusters in the simulation script is enabled by changing the working directory of the OACIS computing host. This enables execution of the simulation using the computer cluster.

– **Simulation script**
  The agent program and the map and scenario used for the simulation are transferred to each computer using SSH. Then, after executing the agent's compilation script, the simulation is started. This script also controls the activation timing of simulators that are necessary for starting the simulation. If the script ends normally, exit code 0 is outputted. Otherwise, if abnormal termination occurs, an exit code other than 0 is output so that OACIS can judge whether the termination was normal. This enables RRS execution using OACIS.

– **Simulator management**
  Various kinds of parameters according to identifiers of agents, parameters of the agents, map, disaster scenarios and so on can be registered as Simulator in OACIS. A unique pattern of parameters according to OACIS Simulator can be registered as ParameterSet in OACIS. OACIS ParameterSet includes the following values which the simulation script and each agent can read.
  - Agent program code
  - Agent program code + Various modules
  - Parameters of algorithms

4 Evaluation

4.1 Purpose of evaluation

With regard to the developmental and experimental support environment proposed in Sect. 3, we developed and tested the agent program at an actual workshop to evaluate whether the re-usability of the code and the burden of the experiment were improved.

4.2 Experimental method

At the workshop of the RoboCup Simulation League held at Fukuoka University on October 22–23, 2016, roughly 30 people were divided into six teams to develop agents using ADF. We conducted an experiment to collaborate and work in combination with AmbulanceTeam, FireBrigade, and PoliceForce agents that were developed by each team. In doing so, we confirmed that it was possible to combine agents developed by different developers in this experiment. At the workshop, we experimented with only six combinations because of time restrictions. However, at a later date, we also experimented with a total of 216 combinations.

Table 1. Top eight results of combined experiments (* are reference values)

Rank	AT	FB	PF	Score
1	A	C	A	143.4091
2	A	B	A	141.4178
3	E	B	A	140.9280
4	A	B	F	136.5078
5	E	B	B	134.8257
6	B	B	D	134.3782
7	C	B	D	134.3404
8	B	B	E	133.8126
	Sample	Sample	Sample	114.2580
	A	A	A	114.2574
	B	B	B	126.5861
(*)	C	C	C	124.3933
	D	D	D	114.2580
	E	E	E	119.9167
	F	F	F	120.8222

4.3 Results and discussion

Table 1 gives the top eight results of the combined experiments of each team (A–F), those of a sample team, and the results for each team as a reference. From the results of the experiments, it can be seen that we arrived at good results even with different teams. In additon, this experiments was conducted easily because it is just changing modules designation that loading. Therefore, we find that agents developed using the framework proposed in this paper can work in cooperation even if they are combined separately.

Moreover, in experimental execution until now, when simulating experimentally in an environment such as a workshop, one operator has basically operated the simulator directly and has managed aspects such as the simulation results. In this case, because there were many steps to manipulate, there were cases in which the results were accidentally not saved. However, no trouble occurred despite many operators being involved in the operation because the operational content was simple and automated. This indicates that the burden on researchers at the time of the experiments can be reduced.

Furthermore, we are planning to hold workshops at universities in Italy, Spain, and France and we will introduce these results and claims to our proposal.

5 Conclusion

In this paper, we proposed an environment as RRS community standard that integrates an agent-development framework and an experiment-management system to support researchers. In the evaluation, we confirmed that code re-usability was improved and that the burden on researchers during the experiment was reduced. We are planning to hold workshops in each country to further confirm the effectiveness of our proposal. As mentioned in Sect. 3.2, the algorithms for solving complex problems (known as the Complex Modules) are aggregated from simpler problems. Thus, in the future, if it is found empirically that the program in question can be divided, we aim to clarify complicated problems as new modules. This will lead to the better resolution of disaster-relief problems. Eventually, we aim to return the research results of the RRS project to society by clarifying disaster-relief problems and proposing individual algorithms that are applicable to disaster relief.

6 Acknowledgment

This work was supported by JSPS KAKENHI Grant Number JP16K00310 and JP17K00317. Y. M. appreciates the support by JST CREST and by MEXT as "Exploratory Challenges on Post-K computer(Studies of multi-level spatiotemporal simulation of socioeconomic phenomena)".

References

1. "RoboCupRescue Simulation", http://roborescue.sourceforge.net/
2. National Institute of Advanced Industrial Science and Technology, "OpenRTM-aist", http://www.openrtm.org/
3. Noriaki ANDO, Takashi SUEHIRO, Kosei KITAGAKI, Tetsuo KOTOKU, Woo-Keun Yoon, "RT-Middleware: Distributed Component Middleware for RT (Robot Technology)", IROS2005, (2005)
4. NTT Data Corporation, "Hinemos", https://ja.osdn.net/projects/hinemos/
5. Yohsuke Murase, Takeshi Uchitane, Nobuyasu Ito, "A tool for parameter-space explorations", Physics Procedia, 57, p73-76, (2014)
6. Takefumi Ohta, Fujio Toriumi, "RoboCupRescue2011 Rescue Simulation League Team Description", RoboCup 2011 Istanbul, (2011)
7. Dai Obashi, Toshiyuki Hayashi, Kazunori Iwata, Nobuhiro Ito, "An implementation of communication library among heterogenous agents NAITO-Rescue 2013(Japan)", RoboCup 2013 Eindhoven, (2013)
8. ECMA, "ECMA-404 : JSON Data Interchange Format", European Association for Standardizing Information and Communication Systems, (2013)

Appendix: The framework and tools

The ADF and experiments support tools on this paper are available on the website(https://maslab.aitech.ac.jp/rrs/rcs17/).