Architecture to Enable Dynamic Reorganization of Cooperative Robotic Teams

Eric Matson
Multi-Agent and Cooperative Robotics Lab
Department of Computing and Information Sciences
Kansas State University, Manhattan, Kansas, USA
etm7766@cis.ksu.edu

Abstract

Cooperative teams of robots must possess the ability to interact. In the event a team member fails or the team begins to operate in a sub-optimal fashion, the team must undergo reorganization with current team members potentially changing roles. To facilitate a cooperative team of robots, physical architecture, software systems and infrastructure must exist to support these teams.

Introduction

Our research goal is to create a generic software architecture that will support cooperative robotic organizations with the flexibility to add new robot instances without changing the existing architecture.

In this paper, we begin with an explanation of the requirements and definitions of organization and reorganization. Then we define the architecture used to implement these ideas within the scope of our research.

Organization/Reorganization

We define a cooperative robotic organization to be a group of robots acting in a specific role to accomplish a goal. An organization must reorganize anytime the structure of the group or the team member's capabilities change.

Reorganization is the process of matching the requirements of the defined goal and the task breakdown structure with the optimal team role configuration to facilitate the most efficient satisfaction of a goal or set of goals. Our model continuously examines the capabilities and availabilities of each member and checks if reorganization is warranted or necessary to optimize execution. The act of organization or reorganization involves the transition from the current operating state of the organization to a more desired state.

The process of organization and reorganization, for an autonomous team of robots, is circular, providing the assumption that the goal satisfaction duration is infinite. Figure 1 exhibits the high-level steps involved in cooperative robotic team organization and reorganization. The initial step is to define the goal of the group to be organized. The next step is to decompose the overall goal into manageable tasks and a set of roles to accomplish the tasks. At the same time the group must assess its individual and collective capabilities. Once tasks, roles and capabilities are defined, the assignment of player to role is completed. At this point the new organization initiates action to satisfy the team goal(s).

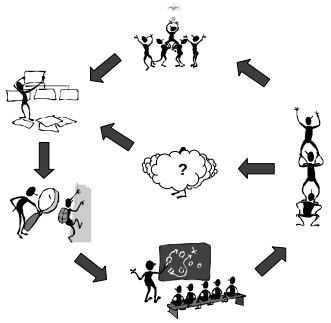

Figure 1: Organization/Reorganization

To provide structure for the organization the formal organizational model is introduced in Figure 2. All elements are shown with the relationship between each element to another.

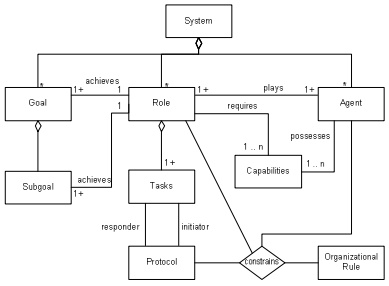

Figure 2: Organization Architecture

System

The system consists of instances of all related components working together to form a set of agents with the ability to form an organization and reorganize itself when required.

Goals

Goals are abstract entities that often must be decomposed to have deliverable outputs. Figure 3 describes a goal structure decomposing a high-level goal of *Rescue Victims* into two sub-goals and consequently seven tasks. Goals are used to identify the critical aspects of system requirements. Therefore, an analyst should specify goals as abstractly as possible without losing the essence of the requirement. This abstraction can be performed by removing detailed information when specifying goals. For example, to "Detect invalid sonar pings" is a goal. How to detect invalid pings is a requirement that may change with time or between various operating systems and is not a goal.

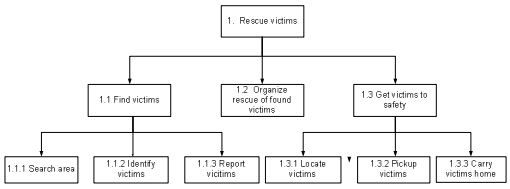

Figure 3: Goal Decomposition Structure

Sub-goals

Sub-goals are goals decomposed into increasingly granular pieces. They are less abstract than goals and take on additional specific and definitive refinement.

Agent

Agents are basically equivalent to autonomous robots in this instance. Agents coordinate with each other via conversations and act proactively to accomplish individual and system-wide goals.

Tasks

Tasks are leaf nodes of the organization's goal decomposition structure. They create granular components with minimal abstraction. In a well defined domain, an organization should be capable of accomplishing tasks, given that the inherent capabilities of the actors or agents is sufficient and they are available to do so.

Protocols

Abstract definitions describing how agents within the organization communicate is defined by the protocol component. It is important to discern the abstract communication model from the actual implementation protocol, with this project, because of the heterogeneous nature of the actors playing in our research.

Capabilities

The robots are defined by the physical and computational capabilities they specifically possess. The robots capabilities define what role they can play in meeting a team goal. For robots, there are two levels of capabilities; computational and physical. The computational capabilities are defined by the level of intelligence built into the robot. The physical capabilities are defined by the range of sensors and effectors included as part of the robot's design.

Role

A role describes an entity that performs some function within the system. In Multiagent Systems Engineering (MaSE), each role is responsible for achieving, or helping to achieve, specific system goals or sub-goals [DeLoach, Matson,Li 2002][DeLoach,Wood, Sparkman 2001]. MaSE roles are analogous to roles played by actors in a play or by members of a typical company structure. The company (which corresponds to system) has roles such as "president", "vice-president", and "mail clerk" that have specific responsibilities, rights and relationships defined in order to meet the overall company goal.

Relationships

Relationships are dynamically allocated, cohesive links that exist from role-to-role, agent-to-agent, and robot-to-robot during the active organization lifespan. The relationships may be based on communication, delegation, cooperation, or other factors.

Organizational Rule

We introduce the notion of *laws* into the organization, which operationalize norms, sanctions/rewards, and their relationship. Laws should also conform to organizational values. Laws are constraints on actions and thus the law (a, s) prohibits the action *a* from being taken when state *s* holds [Shoham and Tennenholtz 1995].

Architecture

To support the development and implementation of an organizationally flexible team of cooperative robots, we designed both hardware and software architectures that work together to form the basic comprehensive architecture to support cooperative team reorganization.

The overall methodology was to look initially for the intersection points of the selected robots to provide a generic robotic interface that will support independent selection of a robot based on its capabilities, not based on the model and physical configuration of the robot.

To create a well-scoped initial project, the functionality base represents common capabilities of the available research robots. The generic programming interface, from an implementation perspective, is constrained by the low level task capabilities of each robot. An example of a low level task capability is the ability to autonomously move about. Another capability is the ability to use sonar for detection, environmental scanning and perception.

We use the development of this organizational model and architecture to create a system to organize and reorganize based on the requirements of the goal, sub-goals and task and the inherent capability set of the available robotic team, to instantiate an optimized organization capable of solving the task at hand. The system will have the ability to choose robots for a particular task based on competence in accomplishing a certain task and the physical and computational capabilities it possesses not the model type. In this manner, the architecture will make decisions without regard to specific hardware requirements.

Hardware

The hardware used in the development of our research begins with the robots available within the Multi-agents and Cooperative Robotics laboratory within the Department of Computer Science at Kansas State University. The robotic hardware used in the prototyping of our initial research architecture begins with the robots included in the first project phase. Four of the robots used are pictured in Figure 4.

Figure 4: Initial Robot Team

The base architecture developed consists of two models of Nomadic Technologies robots, two models of ActivMedia robots and a Parallax Javelin card. The Scout and Scout II are the robot models from Nomadic Technologies and the ActivMedia Pioneer and Amigobot, shown in Figure 5.

Figure 5: AtivMedia Amigobot

Although these models are used as the initial team, the overall scope of the project is to develop a generic model capable of supporting any robotic instance regardless of the proprietary architecture limitations. The initial project architecture will additionally support most robots from ActivMedia and Nomadic such as the Nomad 200 shown in Figure 5. The distinction is that the higher level abilities of some robots will not be supported in the initial design and release of the architecture. The architecture model will capture basic motivational, sensing and effecting skills common to most robots.

Figure 5: Nomad 200

An important measure of a robot is its physical abilities to play a specific role within an organization. Whereas a robot is defined by its computational and physical characteristics and capabilities, we will use the common physical characteristic of sonar to compare the hardware of our robot team.

The Nomad Scout robots, from Nomadic Technologies, have multiple sensors and mainly utilize a sonar ring for environmental sensing. The sonar ring is a Sensus 200 consisting of 16 Polaroid 6500 sonar ranging modules fixed in 22.5º increments in a full 360º configuration. The Polaroid 6500 module can accurately measure distances from 6 inches to 35 feet within a +- accuracy of 1%.[Nomadic Technologies 1999] [Nomadic Technologies 1997].

The ActivMedia Pioneer robot has a configuration of 16 sonar with 8 positioned in the front and 8 in the rear. The front sonar have 3 units spaced 15º away from each other away from center and 1 on each side. The same configuration is used in the back. This configuration provides a full 360º sweep of its environment [ActivMedia 2002].

The ActivMedia AmigoBot has a single array of eight sonar. Six of the eight sonar are positioned in the front of the robot and 2 are positioned in the rear providing a full 360º range of sensing. Two sonar units are positioned 12º away from center, two are positioned 44ºaway from center and the last two, in the front, are positioned 90º away from center. The two rear sonar units are positioned 144º away from center (front) [ActivMedia 2001].

The comparison of hardware capabilities is important to establish baseline capabilities of each robot. A minimal sonar comparison is shown in Table 1. In this case, each robot has a full 360º sweep of its environment, but the differentiating factor is the range that each robot can detect within its environment. Each robot has a lower bound of 6” but the Nomad has a greater ability to detect, with a maximum available range of 35’, than the Amigobot with an maximum range of only 10’.

	Sonar Sweep	SonarRange
Nomad	360	6” – 35’
AmigoBot	360	6” – 10’
Pioneer	360	6” – 21’

Table 1: Hardware Comparison

The applied reasoning, within an organizations perspective, suggests that the Nomad has a higher capability to perform some functions requiring sonar than the AmigoBot or the Pioneer. This will provide a baseline level of reasoning to determine basic capabilities for organization and reorganization based on individual robotic physical abilities.

Operating System Architecture

Operating systems support is critical for designing and evaluating the portability of our software architecture. The first step is to determine what systems will be required to implement the architecture and the second step is to determine the target platforms supported.

The Nomad robots operate on a number of operating systems and hardware platforms, including most Unix and Linux distributions as well as being compilable, in a limited sense, for Win32. Red Hat Linux 7.X is used as the operating system for our development and one of the platforms for deployment.

The Pioneer and AmigoBot run primarily on the Linux and Win32 operating system platforms.

Software

The overall design idea was to look initially for the intersection points of the selected robots to provide a generic robotic interface that will support autonomously configurable robotic organizations. This requires independent selection of a robot based on its capabilities, not based on the model and physical configuration of the robot. The organization has to possess the intelligence, through design, to select the most appropriate agent to participate in an organization by matching the organizational requirements with the ability to participate in a role. In the previous section, we discussed the hardware capabilities inherent in each robot. In this section, we will discuss the software choices and design used to implement our architecture.

The initial step to design a generic architecture is to consider the factors existing in the architectures and platforms to be abstracted into a generic platform. In this case we must examine the pre-existing platforms of the Nomadic and ActivMedia robots used in this project.

The Nomadic Robots can either be programmed in C or Lisp and have an open source code base available. The ActivMedia robots have several choices for development; Colbert, Saphira or Aria. We chose Aria because we could access the source code and extend its native C++ code base. It was important for us to access the source code for each robot instance so that we could first understand the actual robot manipulation code and also extend the source for our own purposes.

Java was selected as the language of choice due to portability between platforms, networking capabilities and support for the Java Native Interface (JNI) support of the C and C++ instances that required "wrapping" for each robot model.

The scope of the initial phase of this project is to develop the generic operating platform where an organization is abstracted away from the hardware implementation. The complete design is exhibited in Figure 6. The parts within the scope of this project are the *Lower Level Functionality* and the *Interface Layer*.

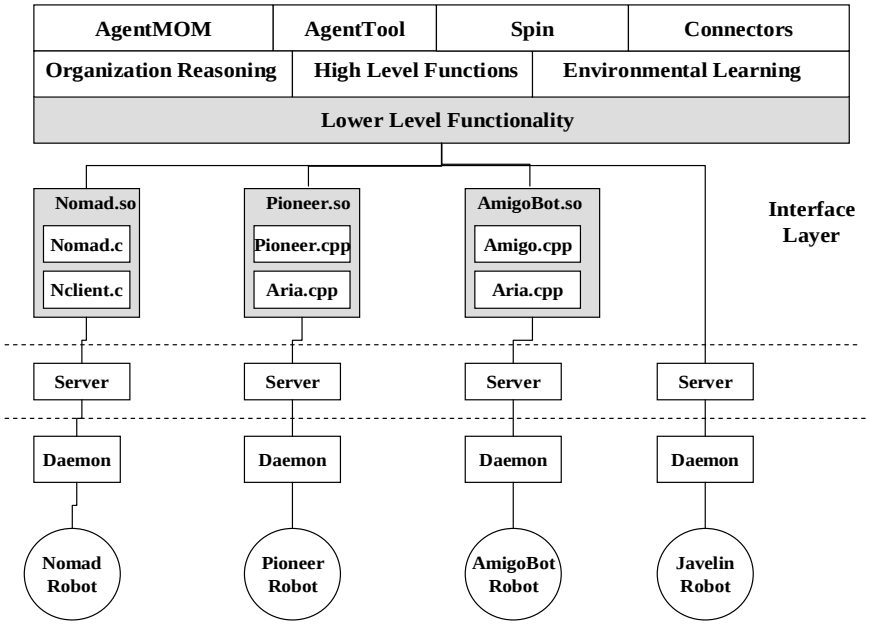

Figure 6: Software Architecture

Lower Level Functionality

This segment was developed by first abstracting basic robotic sensing and movement. Functions such as move, turn, rotate were modeled without regard for the actual implementation by any actual robot interfaces. An example is the move command:

Move(1000)

where the command will make a robot move directly forward 1 meter (1000 mm).

Another example is for rotating around a center point without moving in any direction:

Rotate(180)

where the parameter is in degrees. In this case the robot will rotate 180º and finish with an orientation directly opposite of where it started.

The Lower Level Functions will call interface functions from the Interface Layer. The Lower Level Functionality layer knows

about the robotic instances and which specific interface layer to contact, but hides all of this information from the Higher Level Functions, Organization Reasoning and Environmental Learning components of the architecture.

Interface Layer

This layer connects to the Lower Level Functionality to provide the abstraction interface connection to the robot servers. This layer actually wraps native C and C++ code using the Java Native Interface (JNI).

AgentMOM

AgentMOM is a framework to construct multi-agent systems. It is used as a standard communications protocol between agents to establish conversations. Each agent will have a message handler to receive messages and a conversation object to send messages and communicate with other agents [DeLoach 2000].

AgentTool

AgentTool is an implementation of a Multi-agents Systems Engineering methodology used for the design of cooperative, multi-agent systems. In our case, we used the tool to develop the fundamental components of the organization such as goals, sub-goals and tasks [DeLoach and Wood 2001] [DeLoach 2001].

Spin

The Spin verifier software is included in the AgentTool 2.0 distribution package and will be used for modeling, testing and verifying specifications created in AgentTool to represent the organizational model.

Connectors

This component represents interfaces to systems not yet defined. It has no specific purpose until a system would need to interface the architecture instance and cannot communicate via the AgentMOM module.

Organization Reasoning

The organizational reasoning module will use the structural information from AgentTool to create the organization instance. The organizational instance will control the creation of a team, the team state and will monitor if reorganization is necessary to optimize the execution of the group.

High Level Functions

The High Level Functionality module contains algorithms for mapping, search and other multi-robotic functions, where multiple lower level type functions are assembled to create a higher level capability.

Environmental Learning

Learning from the environment is the highest level function considered in the architecture. The architecture will use reinforcement learning and probabilistic techniques to drive the organization and higher level functions to act in a rational manner.

Implementation

The architecture was successfully implemented using the Nomad Scout Robots, the ActivMedia Pioneer robots and the AmigoBots. The operating systems used to operate the architecture are Red Hat Linux 7.X, Windows 2000, Windows 98, Windows XP Professional and Sun OS. Java version 1.3X was initially used and then we upgraded to 1.4 when it was released.

Conclusions

Although this research is preliminary, the results from research, development and implementation of the Lower Level Functionality and Interface Layer architecture is very promising. We have proven the ability to abstract away the need for a simple organization to know about which robot instance is chosen to perform a specific task.

The architecture was used to implement several successful test groups, although the didn't have the higher level function to reorganize based on sub-optimal conditions, they did successfully execute goals scenarios in the domains of search and rescue, mapping, and inter-robot physical coordination.

There are some issues that surfaced during the initial implementation scenarios and test. There were problems with using threading on top of the communications protocols, but the issues were solved by minimal reconstruction of threading algorithms and re-testing.

All other failures or problems can be attributed to environmental sensing errors such as sonar unit failure. Although not necessarily the fault of the interface layers, we will develop additional algorithms to alleviate these problems.

Future Work

Since we are in the beginning of a long project there is a great deal or work remaining to successfully implement the entire architecture. The following is a list of major areas to research, develop and implement. They are not listed in order of importance or significance.

- Extend the architecture, breadth-wise, to include additional robots families.
- Extend the architecture, depth-wise, to include additional common functionality shared between robotic families abstracting hardware and computational capabilities.
- Test the software with additional and more extensive cooperative robotics scenarios and goal domains.
- Test the architecture with the entire families of Nomadic and ActivMedia robots.
- Complete development of all upper level functionality such as Organization Reasoning, High-Level Functions and Environmental Learning Modules.

References

Shoham, Y. and Tennenholtz, M. 1995. On Social Laws for Artificial Agent Societies: Off Line Design. Artificial Intelligence Vol. 73 231-252.

Nomadic Technologies, Inc. July 12, 1999. Nomad Scout User's Manual, Mountain View, California.

Nomadic Technologies, Inc. March 11, 1997. Nomad Scout User's Manual, Mountain View, California.

ActivMedia. 2001. AmigoBot User's Guide. Version 1.7.

ActivMedia. 2002. http://www.activrobots.com/ROBOTS/specs.htm

Deloach, S. 2000. agentMom User's Manual. Graduate School of Engineering and Management, Air Force Institute of Technology, Wright-Patterson Airforce Base, Ohio.

DeLoach, S., Matson, E. and Li, Y. May 2002. "Applying Agent Oriented Software Engineering to Cooperative Robotics," Proceedings of the The 15th International FLAIRS Conference (FLAIRS 2002). pp. 391 - 396. Pensacola, Florida.

DeLoach, S., Wood, M. and Sparkman,S. June 2001. Multiagent Systems Engineering, The International Journal of Software Engineering and Knowledge Engineering, Volume 11 no. 3.

DeLoach,S and Wood, M. 2001. Developing Multiagent Systems with agentTool. in Intelligent Agents VII. Agent Theories Architectures and Languages, 7th International Workshop (ATAL 2000, Boston, MA, USA, July 7-9, 2000), C. Castelfranchi, Y. Lesperance (Eds.). Lecture Notes in Computer Science. Vol. 1986, Springer Verlag, Berlin.

DeLoach, S. 2001. Analysis and Design using MaSE and agentTool, Proceedings of the 12th Midwest Artificial Intelligence and Cognitive Science Conference (MAICS 2001). Miami University, Oxford, Ohio.