Configuration Management for Multi-Agent Systems

Joseph Andrew
Giampapa
Robotics Institute, Carnegie
Mellon University
Pittsburgh, PA 15213

garof@cs.cmu.edu

Octavio H
Juarez-Espinosa
Robotics Institute, Carnegie
Mellon University
Pittsburgh, PA 15213

ojuarez@cs.cmu.edu

Katia P Sycara
Robotics Institute, Carnegie
Mellon University
Pittsburgh, PA 15213

katia@cs.cmu.edu

ABSTRACT

The vision of the Internet as a tremendous pool of compu-
tational resources is fairly dim relative to its recognition as
a great knowledge base of limitless proportions. The reason
can be succinctly summarized as being due, chiefly, to the
problems of describing and locating large numbers of compu-
tational resources suitable for hosting equally large numbers
of diverse agents or software components. It is also due to
the seemingly–limitless time and energy required to phys-
ically install all of the software and libraries on which the
software depends, on the target systems. Still, a third aspect
of the difficulty is the problem of any one administrator to
properly monitor the runtime status of his agents and the
constantly-changing availability of the platforms on which
to run the agents. The solution of all of these problems
put together is what we call multi-agent system configura-
tion management. In this paper we present the RETSINA
Configuration Manager, *RECoMa*, which is our first step at
addressing this type of scalability problem. [1]

Categories and Subject Descriptors

D.2.9 [**Management (K.6.3, K.6.4)**]: Software configura-
tion management; D.2.9 [**Management (K.6.3, K.6.4)**]:
Life cycle; K.6.4 [**System Management**]: Centralization /
decentralization

General Terms

Management, Design, Theory

Keywords

Multi-agent systems, configuration management, distributed
computing, policy, process, deployment, component, launch
order dependency, runtime dependency, heterogeneous, open,
unpredictable, RETSINA, RECoMa

1. INTRODUCTION

It is generally acknowledged that the Internet is the great-
est knowledge base of limitless proportions that ever existed,
and will continue to grow. But hardly anybody is ready to
identify it as the greatest pool of computational resources
that ever existed, despite its continued growth. That is be-
cause nobody can imagine the process of installing millions
of different programs on millions of different computers as
any activity that can be accomplished in any useful time.
The problem and the vision of how to achieve this scale of
resource allocation is what we refer to as the problem and
the vision of configuration management [CM] for multi-agent
systems [MASs].

Part of the problem of configuration management for open,
multi-agent systems has its origins in what is called the
connection problem[4] — finding the computer with the re-
sources that match an agent's configuration requirements.
Inherent in its solution are the difficulties of how to repre-
sent the descriptions of the computational resources offered,
those required, and the algorithms and locations of the ser-
vices that will help one perform such matches on a world
wide scale. Such concerns have been the subject of study
for multi-agent systems of much smaller dimensions in [14,
13, 15].

Another facet of the problem is how to empower a single
person to control the programs on a wide variety of dis-
tributed and remote computers from a single point of com-
mand. In consideration to the complexity of attempting to
launch many hundreds of agents on heterogeneous, remote,
and distributed computers, there are the problems of how to
check for correct resource allocation matches cheaply and ef-
ficiently, before risking to crash the destination computer or
any of its services. Finally, once the programs are launched
on the myriad of remote machines, there are the problems
of how to monitor the status of the launched programs and
how to track the availability of the computational resources
as they either become available, or are no longer available,
on the target platforms.

In this paper we present the RETSINA Configuration Man-

ager, *RECoMa*. RECoMa is a prototypical system that attempts to address the above problems in terms of their functionality: how to locate, describe, and match resource descriptions. And, although it is not yet possible to access millions of machines on which to test launching millions of programs, RECoMa is capable of controlling and monitoring the status of a fair number of agents on a fair number of computing platforms.

2. THE RECOMA GOALS

RECoMA is an ongoing project that has a set of long term goals. Some of those goals consists of:

- Minimize the time consumed in configuration tasks. The process of allocation of software components in across local and Internet is time consuming when there are not procedures and software tools to support those tasks.

- Minimize the number of errors. The process of configurating a system with many components in a network of heterogeneous computers can be affected by human errors.

- Increase the quality of service is a goal computer systems. With a better control of resources and services the quality of service provided for the system might improve.

- Define policies and process to manage software and hardware items. Those policies will support software platforms to manage MAS applications.

- Provide an information system to MAS managers to minimize memory overload while they create and monitor an application with many agents.

- Provide means to user to balance the load in the computers to increase the efficiency of resources usage. The use of resources includes local networks and the Internet.

3. THE RECOMA ARCHITECTURE

ReCoMa is a software prototype that makes easier to users the tasks of creating and launching MAS applications. We wanted to focus the software in the users. The design of this toolkit is based on the following set of user tasks. To create a MAS application a user must perform the following tasks:

- Domain analysis is the data model of the application. For example, in a manufacturing facility, the software analyst describes entities, relationships between entities, draws flowcharts of the activities performed by the domain experts

- The main task is divided into subtasks that can be performed by agents in the system. The granularity of the division is not well defined but might be based in the existing agents.

- The user selects agents to perform the tasks. To achieve the task the user needs to read the agent profile, which describes the agent requirements, dependencies, and the functionality.
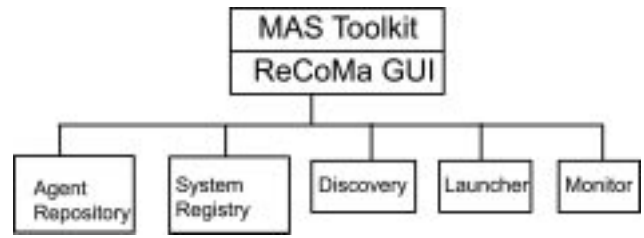

Figure 1: The RECoMa Architecture

- The user allocates agents in the hardware platforms based on the agent requirements, which are described in the agent profile. The agent profiles are compared to platform profiles. The user allocates the agent in one machine when they are compatible.

- Monitor infrastructure agents. For example, a user can display the naming services, and matchmakers available in the agents platform.

- Monitor agents in the platform and the services advertisments.

- The user is able to shutdown the MAS application from the main console.

- Create, edit, and delete new agent profiles in the system.

ReCoMa does not supports the domain analysis and it was designed to support MAS managers instead of end users. The RECoMa architecture, as illustrated in figure 1 has five components the: Agent Repository, System Registry, Discover, Launcher, and the Monitor.

3.1 RECoMa Graphic Interface

Figure 2 shows RECoMa graphic interface. The interface contains three tabs to separate the configurations view, the ANS view, and the Matchmaker View. The top panel (configuration view) contains two tree controls. The left side control tree contains the agent platform information. The tree control in the right side contains the agent repository. The panel in the bottom contains a table that allows users to see the agents that are going to be launched. This graphic interface allows users to maintain the agents repository. From this component, the user can create, delete or update agent profiles. Users can also update the information about the machines connected to the network by sending a registry query. The components of this graphic interface are explained in detail in the following paragraphs.

3.2 The Agent Repository

The agent repository is a database of agent profiles, which contain metainformation about the agents installed in the agents platform. Figure 3 show the information included in the agent profile used in the current prototype. A manager creates an agent profile to register new agents in the repository, when he/she installs an agent provided from external groups or vendors. The agent respository and the agent profiles are represented in the system using XML. The use of XML to represent the repository objects makes RECoMa a
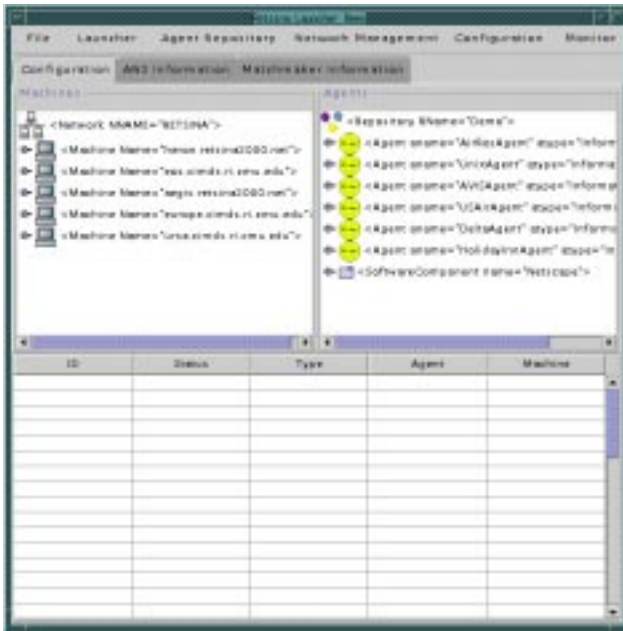
Figure 2: The RECoMa Graphic Interface

system easy to modify. Also the information used by RE-
CoMa is easier to share by using standard formats.

The graphic interface provides operations to add, delete, or
edit profiles. A user working in the graphic interface does
not need to know the XML syntax. Figure 2 shows some of
the data included in the agent profile consists of:

- shell command to launch the agent in the host ma-
  chine. This information includes the command string
  along with the parameters used to execute the agent;

- hardware constraints to execute the agent. For exam-
  ple, the resolution needs to display the graphics pro-
  duced by the agent;

- execution dependencies are included in the profiles.
  For example, some agents need a browser to be ex-
  ecuted; and

- the use of discovery is specified in one slot. This in-
  formation is used by RECoMa to assign values to pa-
  rameters such as ANS and Matchmaker servers.

3.3 The System Registry

The system registry is a database with information about
the software installed in each machine. The machine reg-
istries are necessary to verify prerequisites and dependencies
of a agents before allocating them in one machine.

The system registry is a local database in every machine.
Such database contains the following inputs: environment
variables, java variables, retsina agents, other agents, and
support software. Every agent installed in the local machine
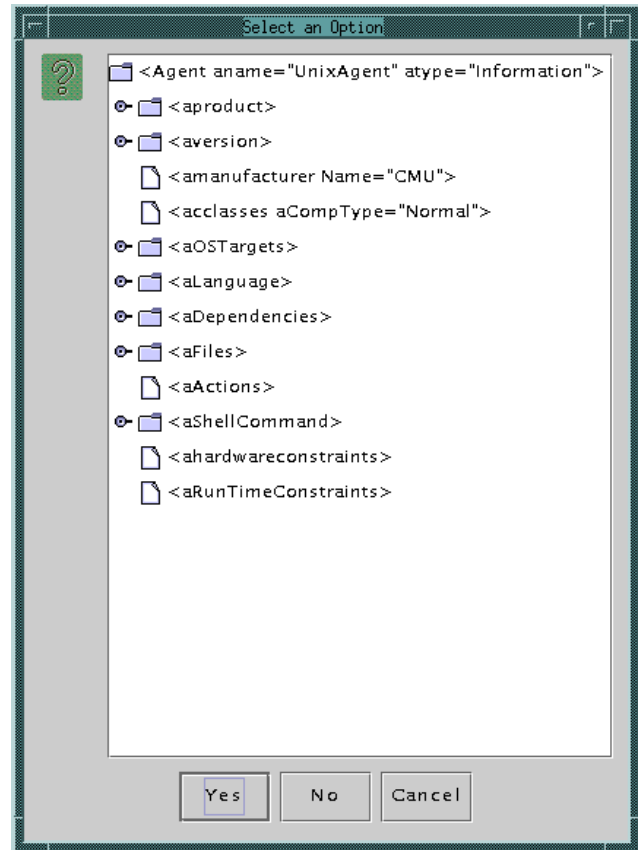must be registered in the registry.

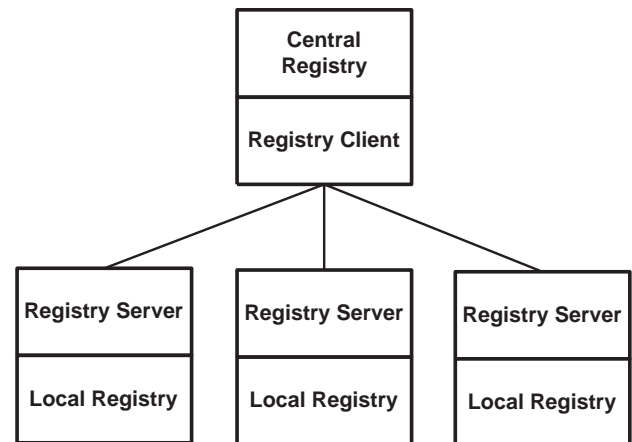
Figure 3: The Agent Profile

Figure 4: The Registry Architecture

The registry is implemented using a client-server architecture as can be seen in Figure 4. The central registry is updated on demand. A server is installed in each machine to answer RECoMa queries. When the server receives the query executes a program that updates the machine profile and send it back using XML as representation. RECoMa executes a client program that sends queries to every machine in the network and displays the information for the user.

3.4 The RECoMa Launcher

A configuration is defined in this work as a set of pairs agent-machine. The search space of configurations is large when it is considered the number of combinations of machines and agents. However, only a subset of the total set of configurations is valid. A configuration is valid in this work when there is not conflict between the agent requirements and the machine profile.

The launcher operations are classified in two groups: the configuration management and the launcher.

Configuration Manager

The configuration manager allows users to create new configurations based on the information presented in the graphic display about agents and machines. Before launching a MAS system the launcher verifies that the agent descriptor and the registry in the local machine satisfies the constraints written in the agent prerequisites, and software dependencies.

The user can save agent configurations to use them latter. When the system loads a configuration saved previously: RECoMa first verifies with information in the registry and the agent the repository to provide feedback to the user about the configuration consistency. The system provided feedback to the user rendering with different color the components that are not consistent in the configuration. The user does need to read the complete set of information in the agent profile and the machine registry to allocate the agent to one machine. The GUI provides visual feedback to users when one agent matches a machine registry. When the agent and the machine match in their requirements and features, they are rendered with a different icons to provide feedback to users. Users only need to read the agent and machine profiles when they have to decide between two profiles that match.

The Launcher

The architecture of this component is based also in the client server pattern. Currently two protocols are designed to launch a valid MAS configuration. The two protocols are XML-RPC (XML remote procedure call) and RMI (remote method invocation).

Figure 5 displays the model used to support the agents launching once the MAS configuration was built using the GUI. Every computer in the network is running two servers, one to support XML-RPC and one to support RMI. The launcher runs the client and requests the machine to launch the agents and other components. The client in the launcher
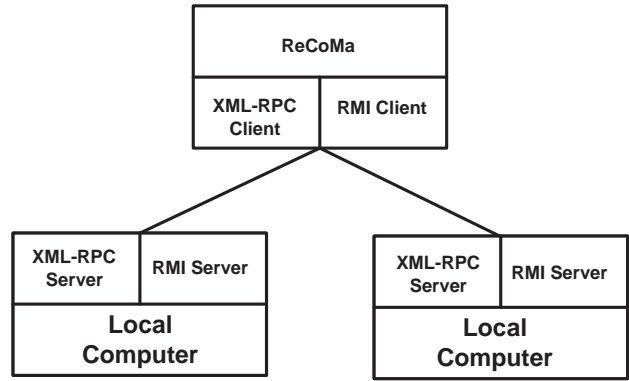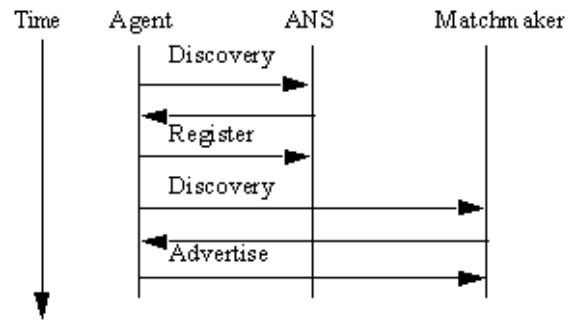

Figure 5: The RMI and XML-RPC Client/Server Model

Figure 6: Messages Exchanged by Agents Using Discovery

builds an XML with the request to the remote machine. The requests are sent from the client to the servers as XML files. The server side in the XML-RPC and RMI protocols interprets the parameters used in the request and translates the XML in actions in the host machine. The actions performed include changing environment variables and to execute software components located in the local machine.

Agents that start from a local computer without being launched from RECoMa work together with agents launched form the central desk; however, they are not registered in the RECoMa configuration. Future versions of RECoMa will register those agents.

The system provides feedback to users when an agent could not be started.

4. DISCOVERY

Discovery is used to provide location transparency to end-users. Instead of launching an agent with the ANS and other infrastructure services as parameters the agent discover them. The MAS manager does not need to know in advance the infrastructure services to launch agents.

Launching an agent uses discovery in order to find the directory services used in the platform. Every RETSINA agent uses discovery in order to find services in the platform.

Figure 7: ANS Services Monitor

Figure 6 shows the messages exchanged when and agent is executed. When the agent does not find services available finish its execution. As can be seen in Figure 6, this agent is looking for two services (ANS and Matchmaker). When the services grow up the number of messages exchanged is larger.

RECoMa useds discovery to obtain the whole list of infrastructure services in the system. RECoMa uses the information obtained to launch agents which do not support discovery. RECoMa uses discovery services are to update the information about new infrastructure services in the system.

Discovery also makes the infrastructure services more tolerant to faults. When a server goes down, every agent has a list of servers in the discovery cache to register with and of them. However, in the remote case that all the services are down the agent can try to discover again the services needed.

4.1 Monitoring

RECoMa provides monitoring functions in order to manage the MAS applications during run-time. The user can see every agent running in the agent platform. However, users can use filters to see only some categories of agents. For example, they might decide to see only the information agents instead of the complete set of agents.

Figure 7 displays the agent monitor interface. This interface contains a tree with the agent name services (ANS) available and a table that allows users to see the agents registered in every ANS. Also the manager can inspect the matchmakers to see wether they are working properly. To get information about the agents running the local agent platform, RECoMa queries the infrastructure agents services available. To get information about the hardware platforms performance some entrees in the registry will be created and the launcher can display to the end user the load in every machine in the network.

5. PRELIMINARY EVALUATION

The current prototype allows managing MAS application at creation time and at run-time. However, more work will be done to cover these two stages under the umbrella of configuration management. The configuration management of MAS will include development, deployment, installing, and run-time management.

6. RELATED WORK

Although there exist a fair number of multi-agent system implementations that solve configuration management problems of other domains[1, 2, 6, 7, 8, 10, 11, 16], ironically very few systems attempt configuration management for the MASs, themselves. The literature that does begin to address CM for multi-agent systems does so either in the context of communications network management agents organized in two tiers[8], or as a proposal for distributed service management in the absence of a particular MAS architecture[12]. Nor is there much specific help from the software engineering community[9, 3] and in the proceedings of a recent configuration management conference[5].[2] Historically, configuration management emerged from the communications world, and is consequently very biased towards the performance-oriented management of very simple hardware devices or connections[12] [3]. For multi-agent systems, however, the requisites and desiderata for configuration management are very different when considering the existence of agent communication languages [ACLs] with context-specific semantics, middle and infrastructure agents[4], discovery services[5], logging and visualization services[6], and the reality that they are implemented in a variety of languages and protocols, and need to operate in a variety of computing environments.

7. FUTURE WORK

A control language will be developed to allow agent applications to reconfigure dynamically. For example, the language can help coordination of the agents under the events of failures.

Another future research topic will be the creation of standard representations of machines and agents to allow RECoMa to be used with different agent frameworks. The current model will be evaluated with users trying to integrate applications using MAS technology. Launcher users will create and launch an application. The time used to solve the task will be evaluated along with the quality of the user solution.

Future research includes designing policies for management of multiagent systems based on agents and components. The dependences and prerequisites model will be evaluated based in how well they supported consistent MAS applications.

Models to obtain more information at run-time will be developed; however, the system must avoid minimizing the system performance by sending a large number of messages over the network.

Another addition to the current prototype includes a module that allows users to create their own agents when agents in the repository are not available or when they try to wrap an agent inside a RETSINA agent. The agent should be created interactively allowing users to select the protocol to use and the coordination paradigms.

---

[2]Not sure how to place Breugst and Choy.
[3]SNMP, CMIP, etc.
[4]ANS, MM papers
[5]SSDP, SLP, our ANS paper
[6]Martin's paper

8. CONCLUSIONS

In conclusion, we identify and characterize multi-agent system configuration management as a distributed computing configuration management problem that is unique to multi-agent systems both in the number and type of figures that the process must encompass and in the life cycle, granularity, and heterogeneity of the components to which it must be applied. As interest in multi-agent systems increases, the resolution of the MAS configuration management problem will also become more critical. After many years of experience and attempts at solving this problem, we feel that we have properly characterized it and provide a road map by which future MAS CM may be pursued.

9. REFERENCES

[1] E. Bodanese and L. Cuthbert. A multi-agent channel allocation scheme for cellular mobile networks. In *Proceedings of the Fourth International Conference on MultiAgent Systems (ICMAS 2000)*, pages 63–70. IEEE Computer Society, 10-12 July 2000.

[2] M. Calisti and B. Faltings. A multi-agent simulator for service demand allocation problems. In *Proceedings of the Fourth International Conference on Autonomous Agents (Agents 2000)*, pages 169–170. Association for Computing Machinery (ACM SIGART), 3-7 June 2000.

[3] S. B. Compton and G. R. Conner. *Configuration Management for Software*. Van Nostrand Reinhold, New York, New York, 1994.

[4] R. Davis and R. G. Smith. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, 20(1):63–109, January 1983.

[5] J. Estublier. System configuration management. In *9th International Symposium (SCM-9)*. Springer-Verlag, September 1999.

[6] K. Hirayama and M. Yokoo. An approach to over-constrained distributed constraint satisfaction problems: Distributed hierarchical constraint satisfaction. In *Proceedings of the Fourth International Conference on MultiAgent Systems (ICMAS 2000)*, pages 135–142. IEEE Computer Society, 10-12 July 2000.

[7] M. Kohli and J. Lobo. Distributed action plans in an agent based network management system. In *Proceedings of the Fourth International Conference on Autonomous Agents (Agents 2000)*, pages 140–141. Association for Computing Machinery (ACM SIGART), 3-7 June 2000.

[8] J.-L. Marzo, P. Vila', and R. Fabregat. Atm network management based on a distributed artificial intelligence architecture. In *Proceedings of the Fourth International Conference on Autonomous Agents (Agents 2000)*, pages 171–172. Association for Computing Machinery (ACM SIGART), 3-7 June 2000.

[9] R. S. Pressman. *Software Engineering: A Practitioner's Approach, 4th ed.* McGraw-Hill Companies, Inc., New York, New York, 1997.

[10] K. Prouskas, A. Patel, J. Pitt, and J. Barria. A multi-agent system for intelligent network load control using a market-based approach. In *Proceedings of the Fourth International Conference on MultiAgent Systems (ICMAS 2000)*, pages 231–238. IEEE Computer Society, 10-12 July 2000.

[11] M. G. Rubinstein, O. C. M. B. Duarte, and G. Pujolle. Improving management performance by using multiple mobile agents. In *Proceedings of the Fourth International Conference on Autonomous Agents (Agents 2000)*, pages 165–166. Association for Computing Machinery (ACM SIGART), 3-7 June 2000.

[12] M. Sloman. Management issues for distributed services. In *Proceedings of IEEE Second International Workshop on Services in Distributed and Networked Environments (SDNE 95)*, pages 52–59. IEEE Computer Society, 5-6 June 1995.

[13] K. Sycara, K. Decker, and M. Williamson. Middle-agents for the internet. In *Proceedings of International Joint Conferences on Artificial Intelligence (IJCAI-97)*. IJCAI, Inc., 23-29 August 1997.

[14] K. Sycara, M. Klusch, S. Widoff, and J. Lu. Dynamic service matchmaking among agents in open information environments. *Journal ACM SIGMOD Record (Special Issue on Semantic Interoperability in Global Information Systems), A. Ouksel, A. Sheth (Eds.)*, 28(1):47–53, March 1999.

[15] H.-C. Wong and K. Sycara. A taxonomy of middle-agents for the internet. In *Proceedings of the Fourth International Conference on MultiAgent Systems (ICMAS 2000)*, pages 465–466. IEEE Computer Society, 10-12 July 2000.

[16] M. Yokoo and K. Hirayama. Algorithms for distributed constraint satisfaction: A review. *Autonomous Agents and Multi-Agent Systems*, 3(2):185–207, June 2000.