SACI: A Tool for Implementing and Monitoring Communication between Agents

Jomi Fred Hübner1 and Jaime Simão Sichman1

Intelligent Techniques Laboratory Polytechnic School of the University of São Paulo Av. Prof. Luciano Gualberto, 158 05508-900 São Paulo, SP {jomi,jaime}@pcs.usp.br
http://www.lti.pcs.usp.br

Abstract

SACI is a tool that simplifies the programming of communication among distributed agents. SACI provides two main types of features: an API for composing, sending, and receiving KQML messages; and tools to relieve agent developers of some difficulties related to distributed environments (agent name service, directory service, remote launching, communication debugging, etc.). This paper explains the fundamentals and usage of SACI. It also demonstrates that SACI has good performance by comparing it with other similar tools.

Keywords Agent Communication, KQML, Tools for MAS development

1 Introduction

The implementation of some types of Multi-Agent Systems (MAS) requires that agents be distributed across a computer network and able to communicate in this environment [11]. However, implementing these systems is not always a simple task: the designer, generally concerned with adding intelligent attributes to agents, also has to worry about problems related to distributed programming (TCP/IP protocol ports, RMI, CORBA, etc.).

Despite the wide variety of tools available to assist in this task, several difficulties are encountered in their use. Among them are:

1. Low interoperability: only a single programming language is supported, or the tool can only be used in a single operating system, etc.
2. Low performance: this fact makes its use in systems with a lot of communication between agents impractical.
3. Difficult to use: installation, configuration, and use are complex and end up requiring the programmer to have knowledge of distributed programming, network protocols, etc.
4. Lack of robustness: the environment does not function properly for a long time.

These difficulties motivated the development of SACI (Simple Agent Communication Infrastructure). SACI is a tool that makes the aspects of distributed communication transparent to the agent programmer, with the main objectives of being simple to use and having good performance.

This article presents the architecture of the SACI development environment, the programming of agents using SACI, and the results of comparing SACI with other tools that have similar purposes.

2 Agent Communication

In a MAS, agents are not developed to solve a specific problem; on the contrary, given a problem, a group of agents comes together to solve it [1]. Thus, the agents do not necessarily need to have been developed using the same tools and models. This group of agents, however, must agree on the use of a common language. Moreover, as in human societies, communication is more efficient when its users follow certain rules, called communication protocols. Some Agent Communication Language (ACL) specifications have been proposed to enable communication between agents developed in different projects and to allow the description of the rules that govern the flow of communication between them. Among the existing ACL proposals, two stand out: KQML and FIPA ACL.

2.1 Knowledge Query and Manipulation Language (KQML)

KQML is a specification for an agent communication language and protocol [7, 8] that aims to be a common means of exchanging information and knowledge between agents1. This specification was widely adopted because it has some important characteristics in an ACL:

- Any language can be used to write the content of messages (for example: LISP, Prolog, SQL, Portuguese);
- The information needed to understand the content of the messages is included in the communication itself;
- When agents exchange KQML messages, the transport mechanism is transparent, i.e., how the message leaves the sending agent and arrives at the receiver.
- The message format is simple, easy for people to read and for a parser to analyze.

Generally, architectures that use KQML add a facilitator agent to the society. This agent knows who the agents in the society are and which requests each one is capable of responding to. In this way, the introduction procedure between agents is simplified.

(1 The exchange of knowledge and not data is one of the factors that differentiates an ACL from a standard language for communication between objects, like CORBA, for example.)

2.2 Foundation for Intelligent Physical Agents (FIPA) ACL

The FIPA ACL specification is based on the same principles as KQML (speech act theory) and has the same syntax for messages [5]. This specification consists of a set of message types and a formal description of their pragmatics—the effect of messages on the minds of the sender and receiver. In this aspect, FIPA ACL differs from KQML; the semantics of the message types are not exactly the same. In addition to the ACL, FIPA also specifies a notation for describing communication protocols. This notation is exemplified with protocols for service requests, contract networks, and others.

3 SACI

SACI was developed based on the KQML specification, with the following main characteristics:

- Agents use KQML to communicate. There are functions for composing, sending, and receiving KQML messages.
- Agents are identified by a name. Messages are transported using only the recipient's name; their location on the network is transparent.
- An agent can learn about others through a yellow pages service. Agents can register their services with the facilitator and ask it about which services are offered by which agents.
- Agents can be implemented as applets and have their interface on a home page.
- Agents can be started remotely.
- Agents can be monitored. Social events (entering the society, leaving, receiving or sending messages) can be viewed and stored for future analysis.2

(2 These last properties of SACI will not be addressed in this article, but a detailed manual for SACI is available at http://www.lti.pcs.usp.br/saci. Information about the implementation of SACI can also be obtained there.)

3.1 Specification

There is a consensus among MAS researchers that agents exist in an environment and interact with it [4, 6, 11]. SACI provides the social part of this environment: a means for agents to know each other and communicate. One way for an agent to know another is by consulting a list with the names of the agents in the society. However, in some applications, an agent needs to know more than just the identification of other agents; it needs to know the abilities of other agents, what they can do.

The information necessary for mutual knowledge among agents forms the structure of the society. Formally, a state of the SACI society structure is defined by the tuple:

Soc = 〈A, S, l , δ〉

such that

A = {α | α is the identification of an agent that belongs to the society}, S = {σ | σ is a skill available in the society}, l is the language of the society, and δ : A 7 → P(S) is a partial function that maps the skills of an agent, such that δ(α) = {σ | σ is a skill of α}.

For example:

lti = 〈{Jomi, Jaime, Julio, José}, {Java, C, Prolog, Teach}, Portuguese, {Jomi 7 → {Java, Prolog}, Jaime 7 → {C,Teach}, Julio 7 → {Java}}〉

The structure of the society can change over time due to social events:

Soci ⇒ Soci+1 | some social event happened at time i.

The social events that can happen reflect the life cycle of an agent:

- **Entering the society:** The agent gains a place in the society by adding its identification to the society. The entry of an agent α changes the structure of the society as follows: 〈A, S, l , δ〉 i ⇒ 〈A′, S, l , δ〉 i+1 | A′ = A ∪ {α}
- **Announcement of skills:** Optionally, an agent can announce its skills to the society. If an agent α announces a skill σ, the structure of the society changes as follows: 〈A, S, l , δ〉 i ⇒ 〈A, S′, l , δ′〉 i+1 | S′ = S ∪ {σ} δ′(x) = { δ(x) if x 6 = α δ(α) ∪ {σ} if x = α
- **Sending and receiving messages to other agents in the same society.**
- **Leaving the society:** The agent α that leaves the society loses its identity in this society: 〈A, S, l , δ〉 i ⇒ 〈A′, S′, l , δ′〉 i+1 | A′ = A − {α} S′ = {σ | σ ∈ δ′(x)} δ′(x) = { δ(x) if x ∈ A′ {} otherwise

3.2 Architecture

- **Entering and leaving societies:** As suggested by the KQML architecture, each society has a facilitator agent that maintains its structure: the identity, location3, and services offered by the agents of the society. When an agent wishes to enter a society, it has to contact the facilitator and register a name. The facilitator will verify the uniqueness of this name and associate it with the agent's location. Similarly, when it wants to leave the society, it must notify the facilitator of that society.

(3 In the SACI architecture, some elements were added to the model presented in Sec. 3.1, for example, the location of the agent on the network.)

- **Sending and receiving messages:** Based on the Application Program Interface (API) proposed by the MASENV tool [2], SACI has a component called MBox that serves as an interface between the agent and the society. Its purpose is to make sending and receiving messages transparent. This component has functions that encapsulate the composition of KQML messages, synchronous and asynchronous sending of messages, receiving messages, announcing and querying skills, and broadcasting messages (see Fig. 1).
- **Announcement of skills:** To be better known in society, agents can announce their skills to the facilitator in a manner analogous to the yellow pages service. Thus, when an agent needs a service and does not know the name of an agent capable of performing it, it can request a list of agents with that skill from the facilitator (Fig. 2 illustrates an example of this interaction). In fact, this is just one model available in SACI for presentation, and other forms can be used, such as broadcasting skills.

3.3 Utilization

To demonstrate some of the functions offered by SACI for programming communication between agents, here is a program that implements an agent that has the ability to add and makes this functionality available to its society:

Java

```
1 import saci.*;
2 public class PlusServer extends Agent {
3   public static void main(String[] args) {
4     Agent a = new PlusServer();
5     Config c = new Config();
6     c.set("facilitator.host", "nantes");
7     if (a.enterSoc("Ag2", c)) {
8       a.initAg();
9       a.run();
10    }
   }
11  public void initAg() {
12    try {
13      mbox.advertise("ask-one", null, "math", "X+Y");
14    } catch (Exception e) { ... }
15  }
16  public void run() {
17    while (true) {
18      Message m = mbox.polling();
19      if (m.get("performative").equals("ask-one")) {
20        Message r = new Message("(tell)");
21        r.put("receiver", m.get("sender"));
22        r.put("in-reply-to", m.get("reply-with"));
23        r.put("content", sum(m.get("content")));
24        mbox.sendMsg(r);
25      }
     }
   }
26  String sum(Object formula) { ... }
27 }
```

4 Comparison with Other Tools

To evaluate SACI, a comparison was made with other tools that have similar characteristics, namely: free distribution, communication between agents is done according to one of the specifications seen in Sec. 2, having white and yellow pages services, and using Java as the programming language. The following tools were selected: JKQML version 5.1a, developed under IBM's alphaWorks project [10]; Jackal version 3.1, developed at UMBC [3]; and FIPA-OS version 1.1, developed by Nortel Networks [9]. The first two follow the KQML specification and the last one follows FIPA ACL.

The criteria adopted for the comparison are grouped into two distinct groups: characteristics and performance of the tools. Regarding the characteristics, the criteria are as follows:

c1. Agents can be run as applets on home pages. c2. Have a monitoring mechanism that shows the messages exchanged between agents and the structure of society. c3. Simplicity of use. In this item, two metrics were used: (a) the number of words in the source code of the agents developed for the test; and (b) the number of configuration parameters that had to be changed to perform the tests. c4. Possibility of defining the communication protocols to be followed by the agents.

Regarding performance, the average number of messages exchanged per second between two agents was measured. These two agents function very similarly to those described in Sec. 3.3, i.e., one agent has the role of a server: it announces its service, waits for requests

(performatives of the `ask` type), and responds to them. The other agent has the role of a client: it finds a server through the yellow pages and measures the time required to make n requests. The tests were performed under three different circumstances:

t1. The two agents running on the same Java Virtual Machine (JVM) as threads. t2. The two agents running on different JVMs but on the same computer. t3. The two agents running on different computers.

4.1 Results

Table 1 presents the results of the comparison considering the characteristics of each tool. JKQML does not exactly have the notion of protocols but allows communication between agents to be regulated through conversation policies between agents. However, the manual does not say how to use this capability. Regarding criterion 3, it is evident that one cannot evaluate the simplicity of using a tool solely by the size of the source code and the number of configuration parameters. However, the numbers collected corroborate the subjective evaluation of the authors, i.e., the tools present the following order of simplicity of use (from simplest to most complex): SACI, FIPA-OS, JKQML, Jackal.

Tool	c1	c2	c3(a)	c3(b)	c4
SACI	yes	yes	421	0	no
Jackal	no	no	382	6	yes
JKQML	no	no	1176	1	yes
FIPA-OS	no	yes	950	1	no

5 Conclusion

The SACI environment has proven to be a viable solution for implementing communication and presentation in a MAS. The problems presented in the introduction are minimized: portability is guaranteed through the Java language, tests have shown that SACI's performance is very good, SACI encapsulates various aspects of distributed programming (eliminating the need for the user to perform configurations), and performance over time does not decrease, on the contrary, it increases. Compared to other tools, SACI presents clearly better performance. However, to perform a more accurate comparison and evaluate the reasons for the results, it is necessary to implement the concept of conversation in SACI (which will be done in future versions).

References

[1] ALVARES, L. O., SICHMAN, J. S. Introduction to multi-agent systems. In: MEDEIROS, C. M. B. (Ed.) **Update Workshop on Informatics**. Brasília: SBC, August 1997. v. 16, Chap. 1, p. 1ss.

[2] CARDOZO, E., SICHMAN, J. S., DEMAZEAU, Y. Using the active object model to implement multi-agent systems. In: **International Conference on Tools with Artificial Intelligence**, 5th, 1993, Boston, USA. Proceedings... c1993.

[3] COST, R. S., FININ, T., LABROU, Y., LUAN, X., PENG, Y., SOBOROFF, I., MAYFIELD, J., BOUGHANNAM, A. Jackal: A java-based tool for agent development. In: **Workshop on Tools for Developing Agents**, 1998, Madison, Wisconsin. Working notes... c1998. (http://jackal.cs.umbc.edu/Jackal/).

[4] DEMAZEAU, Y., MÜLLER, J.-P. (Eds.). **Decentralized Artificial Intelligence**. Amsterdam: Elsevier, 1990.

[5] FIPA, Geneva. FIPA 97 specification, part 2: Agent communication language, 1997. (http://www.fipa.org).

[6] FRANKLIN, S., GRAESSER, A. Is it an agent or just a program? A taxonomy for autonomous agents. In: **International Workshop on Agent Theories, Architectures, and Languages**, 3rd, August 12-13, 1996, Budapest, Hungary. Proceedings... Editors MÜLLER, J. P., WOOLDRIDGE, M., JENNINGS, N. R. Lecture Notes in Computer Science, Vol. 1193. Springer, c1997. p. 21–35.

[7] LABROU, Y., FININ, T. A proposal for a new KQML specification. UMBC, Baltimore, 1997.

[8] LABROU, Y., FININ, T., PENG, Y. Agent communication languages: the current landscape. **IEEE Intelligent Systems**, v. 14, n. 2, p. 45–52, March/April 1999.

[9] NETWORKS, N. FIPA-OS, 2000. (http://www.nortelnetworks.com/products/announcements/fipa/).

[10] TSUCHITAMI, H. Java KQML. (http://www.alphaworks.ibm.com/formula/JKQML).

[11] WEISS, G. (Ed.). **Multiagent systems: A modern approach to distributed artificial intelligence**. London: MIT Press, 1999.