A collaborative multi-agent framework for internet-based teleoperation systems

Nader Cheaib, Samir Otmane, Malik Mallem

A Collaborative Multi-Agent Framework for Internet-Based Teleoperation Systems

Nader Cheaib, Atomic Energy and Alternative Energies Commission (CEA), Gif-sur-Yvette, France

Samir Otmane, Informatics Integrative Biology and Complex Systems (IBISC) Laboratory, University of Evry Val Essonne, Evry, France

Malik Mal/em, Informatics Integrative Biology and Complex Systems (IBISC) Laboratory, University of Evry Val d'Essonne, Evry, France

This paper presents a conceptual model of an agent (called Collaborator Agent) intended to design collaborative software architectures based on multi-agent systems. The authors' model combines astutely two research areas: Multi-Agent Systems (MAS) and Computer Supported Cooperative Work (CSCW). The particularity of their approach is the division of the collaborative process into three spaces according to Ellis' 3C model: communication, coordination and production. In their work, the authors extend the 3C model by adding a fourth space: collaboration. Hence, the authors present a model based on four types of agents (collaboration, communication, coordination and production) supporting the whole set of collaborative tasks. The model is used to create the conceptual software architecture of their MAS. The authors apply their conceptual model on the ARITI-C system for collaborative online robot teleoperation. Finally, the authors present a quantitative evaluation of the collaboration process in ARITI-C.

INTRODUCTION

A Multi-Agent System (MAS) is composed of entities called agents that interact together in order to achieve a common goal. An agent, in general terms, is anything that can be viewed as perceiving its environment through sensors, and acting upon that environment through effectors (Russell & Norvig, 1995). Computational MASs consist entirely of artificial agents executing as software programs and running on a computer system. Some researchers have treated the concept of formal descriptions of MAS, while the aim is to assess their properties and formal specifications. In fact, there are formal descriptions of both agent paradigms and interaction

paradigms; however, some formalisms provide representations of both of them. In the literature, as in Wooldridge and Jennings (2005), formal descriptions are also called theories. The MAS theories are rules that constitute a basis for specification (Glaser, 2002; Occello, 2002\), design (D'Inverno, Kinny, Luck & Woolridge (1998), Ricordel & Demazeau (2002)) implementation and verification of MAS (Arlabrosse, Gleizes & Occello (2004), D'inverno, Fisher, Lomuscio, Luck, Rijke, Ryan & Woolridge (1997), Picard & Gleizes (2004)). There are other methodologies and theories dealing with this concept, such as The Multi-Agent Scenario-Based Method (Moulin & Brassard (1996), Moulin & Cloutier (1994)) that is intended to be applied in the field of cooperative work.

Moreover, software agents represent a fundamental way of considering complex distributed systems and societies of cooperating autonomous components. When building such systems, special techniques are required for their design and implementation. In our work, we aim at assisting the development of such systems by providing a model and formalism that can be used to specify the desirable behavior of MAS, where one requirement is to be able to move from specifications of such systems to implementation. The characteristics identified by using a formalism serve to measure and evaluate implementations of agent systems. On the other hand, CSCW is the field of study that examines how technology affects group interaction, and how technology can be best designed and built to facilitate group work. Activities in that domain are known as groupware.

Ellis, Gibbs & Rein (1991) defines groupware as "computer-based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment". While groupware refers to real computer-based systems, this means that the notion CSCW is the study of tools and techniques of groupware as well as their psychological, social and organizational effects. Some questions arise such as how to conceive and develop adequate collaborative work architectures taking into account communication, coordination and production features, and how to evaluate the developed collaborative systems. In this paper, we attempt to answer these questions by putting forward a new conceptual model of an agent called Collaborator Agent (CA) used to develop and evaluate software architectures of collaborative applications. The proposed CA is based on an approach combining astutely two research area: MAS and CSCW (Ellis, Gibbs & Rein (1991), Laurillau & Laurence (2002)). In the first area, a formal model of the MAS (Ferber approach in Ferber & Muller (1996)) is used to bring out the features of the agent. The second area provides a useful specification of collaboration based on the communication, coordination and production theories. Then, we apply our conceptual model on the ARITI-C system for collaborative online robot teleoperation. This system offers web interfaces in order to assist users in collaboration to remotely manipulate real objects via a robot.

We will proceed as follows: In section 2, we present some related work in the field of MAS in many application areas. In section 3, we present the conceptual modeling of the CA. In section 4, the design and implementation of the CA are presented and discussed. In section 5, we present the ARITI-C system for online collaborative robot teleoperation. Furthermore, we present the quantitative evaluation of the system in order to assess the collaboration process. A conclusion and future work are presented in the last section.

RELATED WORK

Various work exist in the literature that make use of MASs in many application domains. In particular, the use of software agents for robot teleoperation has been gaining much attention since many years (Lin, Song & Anderson (2005), where agents can communicate, coordinate and negotiate in order to meet their goals in a framework suitable for task execution. Also, the autonomy and intelligence of software agents have considerably increased software automation of many operational areas. For example,

the authors in Cabri, Ferrari & Leonardi (2004) show an approach based on mobile agents for communication purposes, where a message becomes an active entity providing content filtering, while giving means to the message itself in order to pro-actively decide of what to do with its content, as well as how to present it to end users. This approach can be integrated in existing communication applications, which facilitates its exploitation by users. Furthermore, Selliah, Reddy, Bharadwaj & Reddy (2004) present research work on collaborative groups assisted by agents. They introduce the Eksarva platform, which supports the formal modeling of collaboration as a behavior function as well as workflow rules. This approach is used to schedule agents executing integrated activities related to knowledge management for efficient group collaboration. Frey, Stockheim, Woelk & Zimmermann (2003) propose the use of MAS for supply chain management, while following a few prospects as the organization, control, and execution. This approach incorporates the use of design techniques standards using UML. Cabri, Ferrari and Leonardi (2003) offer an agent-based application in order to analyze different types of agent interactions. These interactions are handled separately from agents' logic using roles. It allows application development based on flexible agents in order to execute automatic tasks.

Many research work have also been conducted in order to implement MAS for robot teleoperation. For example, Lin, Song and Anderson (2005) present a distributed hybrid agent-based control (ARC) architecture for multi-robot cooperative tasks. This software architecture also provides an efficient platform for building up a multi-robot system consisting of heterogeneous robots. In this system, each individual agent is responsible for sensing, actuating, or executing a set of functions, while it can also exchange information with other agents. Furthermore, Wegner and Anderson (2006) describe an approach to multi-robot control for search and rescue environments, where robotic agents are implemented using a schema-based architecture with autonomous behaviors. This approach involves the use of two software agents running on each robot to effectively balance the teleoperated and autonomous components. A mediation agent is used to integrate the commands from a teleoperator with a robot's autonomous processing, while an intervention recognition agent recognizes situations in which an operator should intervene. Moreover, Lewis, B. and Tastan, B. and Sukthankar, G. (2010) present CoOperator, an agent-based human-robot interface that infers operator distraction and identifies any robots that are not currently being effectively managed. The CoOperator interface is designed to assist the human operator in managing, controlling, and navigating multiple robotic agents through a disaster scenario. In this approach, robots are monitored in order to determine which ones are suffering from operator neglect, by employing a Hidden Markov Model in order to infer operator neglect from the robot's observable state. Finally, Cragg and Hu (2003) present a software architecture that makes use of mobile agents' technology for robot teleoperation for nuclear decommissioning. Their proposed architecture contains these main components: User Computer (U) that allows a user to control or supervise one or more robots, Control Agents that determine the current activity and location of robots, Planning Agents that help robots to determine an effective long-term behavioral strategy, and Mapping Agents that allow a consistent Long Term Map to be constructed in all robots, and provide the user with a characterization map developed online.

A common aspect of most of these teleoperation systems is their use of many cooperative robots that are being teleoperated by one operator. They do not take in consideration the collaborative process involving many users teleoperating the same distant robot. Hence, in our approach, we explicitly take in consideration the collaborative aspects between users, by proposing a set of four agents representing each user in the system. Furthermore, we rely one the 3C model for defining the collaborative process involving autonomous agents that are communicating, coordinating and producing

together in order to effectively teleoperate a distant robot. In the following, we describe the conceptual model of our system.

CONCEPTUAL MODELING OF THE COLLABORATOR AGENT (CA)

In this section, we describe the organization of agents as well as their interactions during a collaboration task. Also, we present the conceptual modeling of the CA. To further understand the concept of collaboration, we will first explain the 3C model that we base our formalism upon.

Collaboration Specification

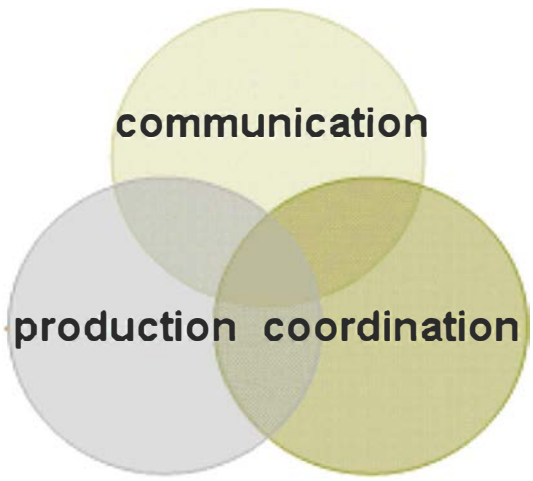
Ellis, Gibbs & Rein (1991) propose classification means of a collaborative system as shown in Figure 1. Hence, a groupware system covers three types of services: communication, coordination and production:

1. The communication space refers to person-to-person communication such as e-mail, relay chat and mediaspaces;
2. Coordination may be viewed as the link connecting the other two C's in the 3C model in order to enforce the success of collaboration (Fuks, Raposo, Gerosa & Lucena (2005)). Therefore, the coordination space covers activities' dependencies including temporal relationships between the multi-user activities. It also refers to the relationships between actors and activities;
3. The production space refers to the objects produced by a group activity or to the objects shared by multiple users. This functional model is known as the 3C model.

The importance of each of the three spaces depends on the nature of the tools considered. Hence, the distinction according to the three spaces is not strict, as it is possible that a coordination activity takes place after a communication activity. For example, making an appointment with someone on the phone is a coordination act based on a communication one. Moreover, the global functional role can evolve in the course of time, while this evolution can be modeled using the 3C model. For example, it is possible that the group activity is centered on communication, while the production system is in use.

There exist some work in the literature that make use of the 3C model in order to construct collaborative applications (Laurillau & Laurence (2002), Fuks, Raposo, Gerosa, Pimental & de Lucena (2007), Oliveira, Antunes & Guizzardi (2007)). Fuks, Raposo, Gerosa & Lucena

Figure 1. Ellis' 3C model



(2005) affirm that the way people connect and communicate with each other has changed through the change of the society over the years. According to the authors, the understanding of communication has transformed from being vertical, where orders are passed from above and reports are sent up the line, to a peer-to-peer paradigm where communication, coordination and cooperation predominate. This is due to the fact that command and control paradigm is losing effectiveness in the society. People are increasingly using tools and applications with no specific or centralized source that issues orders, but where people are collaboratively coordinating and dividing tasks between them, and eventually taking group decisions.

According to Fuks, Raposo, Geros, Pimentel and de Lucena (2007), the 3C model can also help evaluators focus their attention on the communication, coordination and cooperation aspects of the application in order to detect usability problems. Also, the relationship among the 3Cs of the model can be used as a guidance to analyze a groupware application domain. For example, a chat tool can be seen as a communication tool that requires coordination (access policies) and cooperation (registration and sharing). In our work, we use the 3C model in order to define the three main aspects of a collaborative application. We affirm that users need to pass through a communication phase, then a coordination phase and eventually a cooperation or production phase. Hence, an optimal collaboration pattern is achieved when the collaborative process is initiated by communicating, and ends by a concrete realization of the task at hand.

Agent-Based Collaborative Software Architecture

The aim of the proposed model is to assist the collaboration as well as the resolution of conflicts that could result from it. The use of agents enables to create a subtle environment (responsiveness, proactivity) in the goal of a successful execution of tasks, as well as the termination of missions in good conditions. The MAS (CA) presented in this paper is composed of four types of agents:

- Agents that choose a mission, called communication agents;
- Agents that select actions to be executed, called coordination agents;
- Agents that execute the chosen actions, called the production agents; and
- Agents that handle the inter-agents communication, called collaboration agents.

Every instance of these four agents is regrouped under one super agent, called the Collaborator Agent (CA) shown in Figure 2.

These agents are bound by a cement agent, the "collaboration agent". The latter insures the communication, on one hand, between the three dedicated agents, and on the other hand, with other CAs. In our system, a CA interacts with other CAs through its "collaboration agent", as well as its dedicated agents ("communication agent", "coordination agent", "production agent"). Furthermore, these agents directly communicate with their corresponding agents belonging to other CAs, as shown in Figure 3:

- **Collaboration agent:** As mentioned, the collaboration agent insures two principal functions: it allows communication between the three dedicated agents inside the CA, and it establishes a direct interaction between agents belonging to the same class (communication, coordination and production) without the intervention of their collaboration agents. This direct communication between the three dedicated agents, inside a CA, and their corresponding agents increases the system's efficiency (as the communication does not depend on the collaboration agent) in order to avoid the "bottleneck" effect made by centralization;
- **Communication agent:** This agent decides if a mission can be executed or not, and therefore, whether collaboration can take

Figure 2. Collaborator agent (CA)

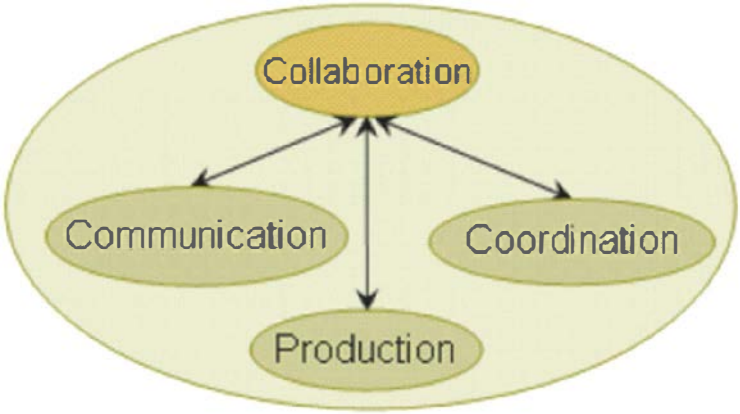
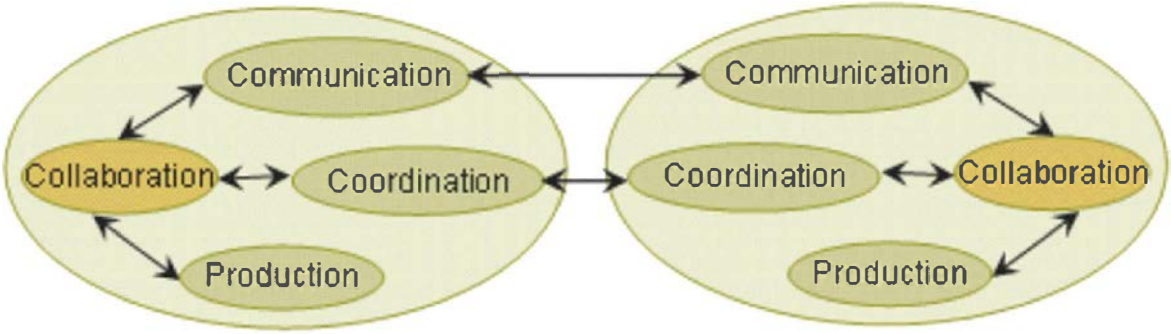


Figure 3. Interaction between two CAs



place. Thus, it is the corner stone of the CA (knowing that a CA is composed of four agents: communication, coordination, production as well as the collaboration agent as the cement agent). The information sent to this agent presents different perceptions from the outside world *Percept{i}*, while producing a set of output that changes its environment:

- *Inf{ij}* is the set of information an agent ***i*** sends to another communication agent *j*;
- *Percept{i}* is the set of stimuli and sensations it generates;

- **Coordination agent:** This agent defines all actions that it can accomplish. The information sent presents the different perceptions that can be received from the outside world *Percept{j}*). It also produces a set of output that changes its environment:
  - *Actions{i}* are the set of executable actions;
  - *Percept{i}* are the set of stimuli and sensations that it gives out;
- **Production agent:** This agent is responsible for executing actions stemming from collaboration agents. The information sent to it presents different perceptions that can be received from the outside world *Percept{j}*. It also produces a set of outputs that changes its environment:
  - *Percept{i}* are the set of stimuli and sensations that it gives out;
  - *Results{i}* are the set of results stemmed from the execution of actions.

We can see a conceptual architecture of the CA in Figure 4.

DESIGN AND IMPLEMENTATION

In this section, we present the implementation of the main systems' classes as well as their interaction mechanisms. First, we present the design of our software architecture through UML diagrams, and then we present the communication protocol used.

Software Architecture

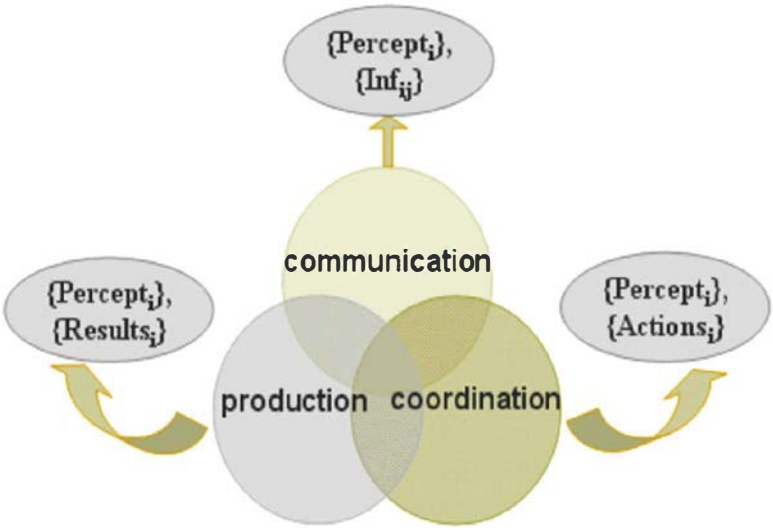
The software was developed using JADE (jade.tilab.com} framework built in Java technology, which gives a complete runtime environment for agents' manipulation. It also provides various utilities, such as the Directory Facilitator (DF) agent, which provides a "yellow pages" service that we used to reference the sub-agents of a CA. We implemented a *GenericAgent* class that derives from the JADE *Agent* class, which provides methods that facilitates the registration process in the DF, as well as the process of message sending. A *Generator* class creates the CAs through four classes *CollAgent, CommAgent, CoorAgent* and *ProdAgent*. Each instance of an agent class includes behaviors deriving from the following JADE classes:

- **CyclicBehaviour:** Simple listener to incoming messages identified by a specified field;
- **AchieveREInitiator:** Initiator for a Rational Effect conversation, as defined by FIPA (http://www.fipa.org/). Such a conversation is characterized by an *Agree* or a *Refuse* message sent by the receiver;
- **AchieveREResponder:** Responder for a Rational Effect conversation;
- **ContractNetInitiator:** Initiator for a *Contract Net* conversation, as defined by FIPA, i.e. a conversation in which a bid (or a Refuse) message is sent by each receiver, allowing the initiator to choose the best offer;
- **ContractNetResponder:** Responder for a Contract Net conversation, generally making a bid for the received offer.

Collaboration Protocol

This section gives a brief overview of the exchanged messages during the three phases of the collaboration process. This protocol might not be the optimal one for the CA model,

Figure 4. Conceptual architecture of the CA



however, it gives an example of the concepts mentioned in previous sections when applied in a practical scenario.

Communication Phase

This step of the collaboration process is divided into two parts: the creation of groups, and the choice of a mission within a group. To create a group, we choose the following scenario: First, every agent is registered in the DF as "free". Collaboration agents randomly ask their free counterparts to join them (*Rational Effect* protocol). Every agent joining the group modifies its registration in the DF as a part of the group. The CA that initiated the group is called the "master" agent, and will supervise the group's collaboration process. When a group is formed, a list of available missions is sent to the communication agents by the collaboration agents. When the mission is chosen, the master communication agent notifies its collaboration agent, which notifies its counterparts in the group. The communication process ends.

Coordination Phase

When a CA receives the mission chosen during the communication process, it sends its coordination agent the chronologically ordered list of actions for the mission. This process is presented as a sequence diagram shown in Figure 5: in this example, two actions are distributed within a group of three agents, the master being number 1. When all the actions are distributed, every collaboration agent has a list of its attributed actions. The production phase can then start.

Production Phase

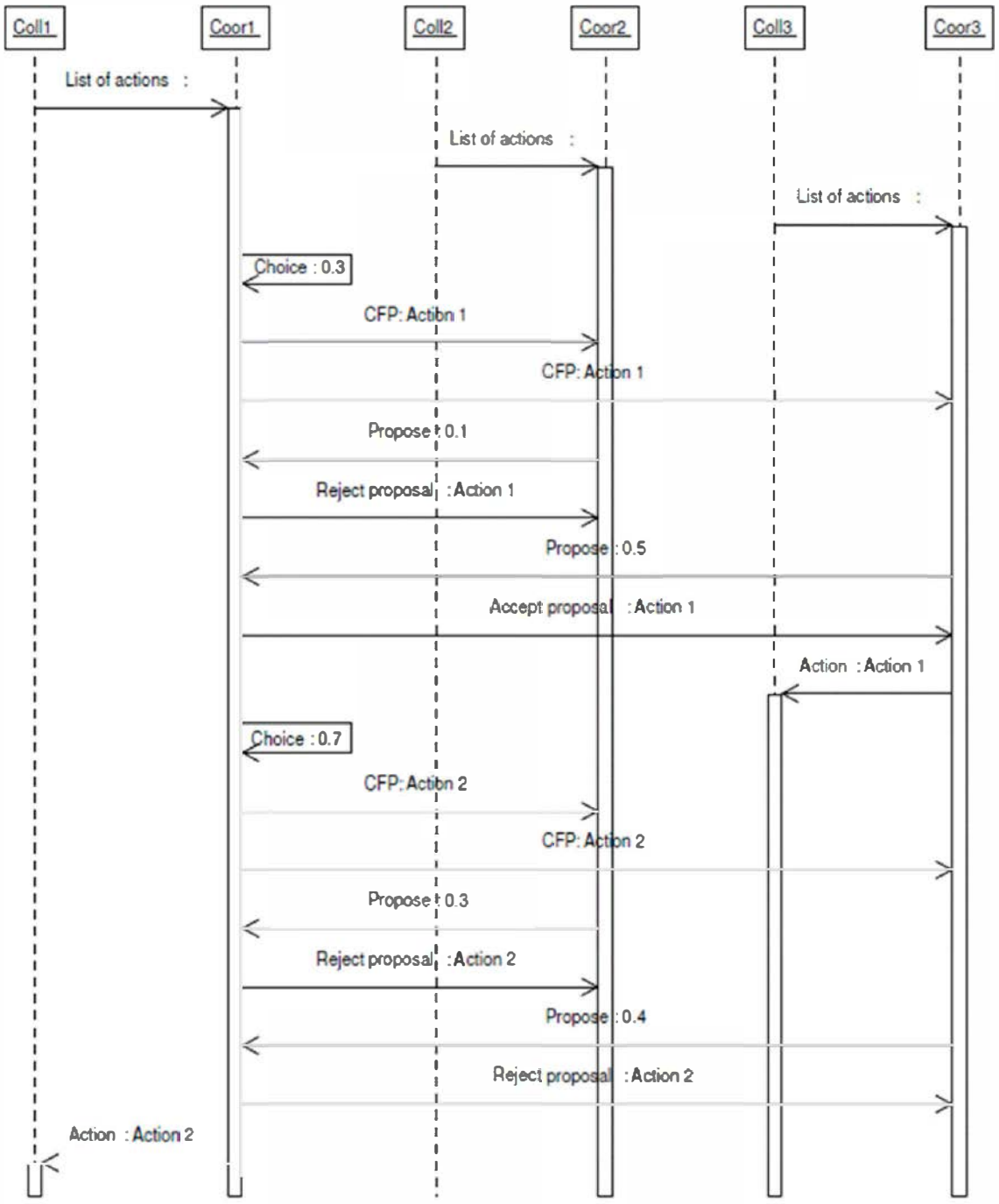
When all actions have been distributed, the master coordination agent sends to its collaboration agent a signal to start the first action, which is in turn sent to all other collaboration agents of the group. Hence, every collaboration agent is notified of the current action to be done. The collaboration agent that has been attributed this action sends its production agent a message to execute it. When the action is finished, the active production agent sends a message to its collaboration agent, which notifies all its counterparts in the group. The master collaboration agent notifies in turn its coordination agent. If any actions remain to be performed, the process restarts (the master coordination agent sends a signal for the next action), otherwise, every agent in the group is notified of the end of the production phase.

In fact, the production process in our system was improved by adding "supervision" capabilities: the active production agent sends to the inactive ones the status of production through the collaboration agents. In a practical application, this would allow co-workers to be aware of a task's status while being performed. Due to the large number of implied messages, this feature was deactivated in our quantitative experimentations. In the next section, we apply our conceptual model on the ARITI-C system for collaborative online robot teleoperation. We finish the section by presenting a quantative analysis of the collaboration process in the system.

FROM ARITI TO ARITI-C: COLLABORATIVE TELEOPERATION VIA THE INTERNET

The objective of the ARITI project (Augmented Reality Interfaces for teleoperation via the Internet) is the study and implementation of new assistance methods for remote and collaborative work. The work concerns, on one hand, the development of interfaces (single and multi-user) through web technologies and, secondly, the development of multisensory and multimodal interfaces using virtual and augmented reality (VR/AR). The application areas of the ARITI project are: telerobotics, telemedicine, computer science, molecular biology, as well as sectors concerned with group activities in order to design, take decisions and analyze complex problems. In fact, initial work concerning the ARITI project allowed the creation of the first teleoperation system in AR via the internet in

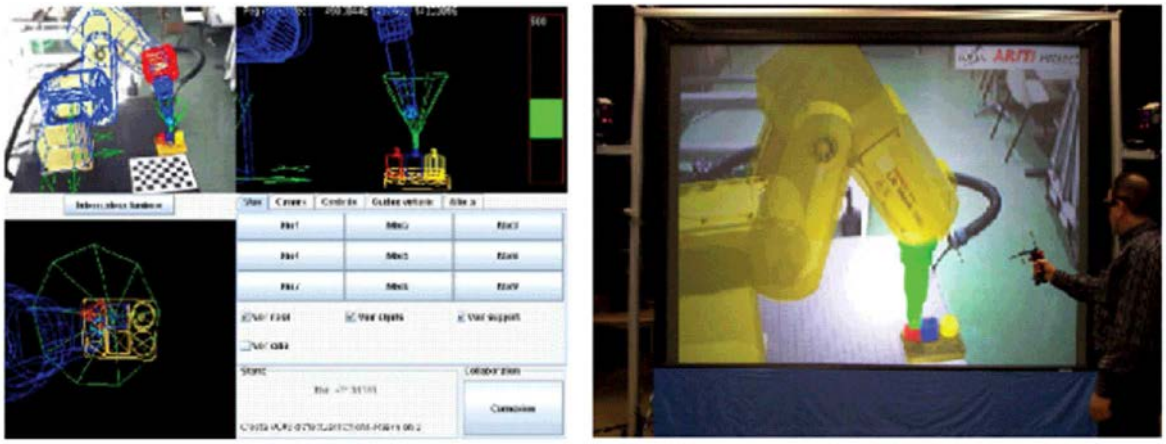
Figure 5. Typical sequence diagram for the coordination step



France (Otmane, Mallem, Kheddar & Chavand (2000)). This system allows assisting a user in remotely manipulating real objects via a robot, through a web interface. The techniques used provided the operator assistance for the perception of the scene in order to control the robot. Hence, the aim is to enhance tasks' precision while insuring the needed security for its execution. A snapshot of the system is shown in Figure 6.

In fact, most of the complex missions are realized either directly on the site (industrial maintenance, design of complex artifacts, delicate surgical intervention, etc.), or from a remote

Figure 6. Screenshot of ARITI (web interface at left, mixed-reality semi-immersive platform at right)



site (teleoperation in hostile environments, telemaintenance, telediagnostics, etc.). However, the success of some missions often requires collaboration with other people (including experts) having various expertise and multiple roles. It is in this context that work has been conducted for the design of a groupware application for teleoperation assistance over the web, which we call ARITI-C (Khezami, Otmane, Mallem (2005)). The objective of this work concerns the design of a collaborative multi-agent system for assisting mission preparation as well as the execution of collaborative tasks. The first real application of this work concerns the collaborative teleoperation of a robot via Internet using the ARITI-C system, based on the multi-agent system proposed in this paper. Hence, the system's interface allows a group of remote users to communicate, coordinate and cooperate before and during the execution of teleoperation tasks. The ARITI-C system is shown in Figure 7.

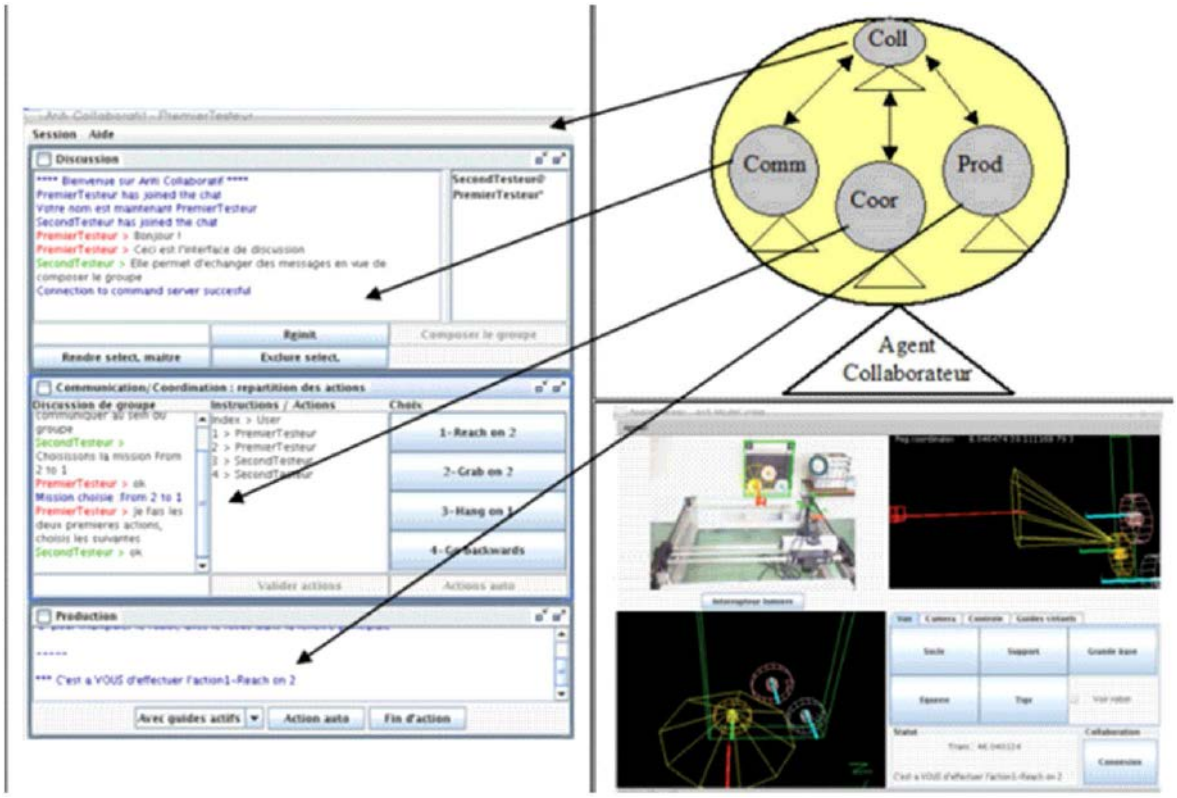
Collaborative Teleoperation

The challenge of collaborative teleoperation via Internet should take in consideration, on one hand, requirements from the teleoperation domain, and on the other hand requirements from the CSCW domain (Computer Supported Collaborative Work) (Ellis, Gibbs & Rein (1991), Ellis & Wainer (1994), Ellis (1999), Coleman & Shapiro (1992)). In fact, this challenge consists of finding a compromise between flexibility that a groupware should offer, and the guarantee of realizing a teleoperation mission, in which the operator needs assistance and security in the task process. In consequence, a collaborative teleoperation groupware via Internet should:

- Support collaborative work;
- Assist and supervise users in the mission process;
- Allows the analysis and evaluation of the collaboration within a group and/or a set of groups;
- Allows the realization of tasks in a synchronous way (real time) and/or asynchronous (differed time) way depending on the mission.

In fact, the studies held in the domain of MAS have two objectives. The first is to present concepts, properties and formalisms for the design of complex and distributed systems. The second objective is to study multi-agent development platforms that can be used for implementing such systems. In the CSCW domain, the study presented attempts to answer the following questions: What definition can we give for the collaboration term? How can we design the collaborative teleoperation, and what software architecture model should be used? In the MAS domain, the agent formalism proposed by Ferber (1997) as well as the

Figure 7. Screenshot of the new interfaces for online collaborative teleoperation via ARITI-C. At left, the assistance interface for communication, coordination and cooperation (production). At right, the Collaborator Agent as well as the collaborative teleoperation interface).



JADE development platform were adopted and used. In the CSCW domain, we adopted the concept of the 3C model for groupware (Ellis & Wainer (1994)):

- **A Multi-Agent system for collaborative teleoperation:** The proposed system has an objective of assisting the collaboration process as well as the resolution of conflicts that may result from it. The use of agents allows to build a flexible environment (agents capable of responding in a specified time, proactive, communicative) in the purpose of an efficient execution of tasks until the end of the mission. In our teleoperation system, a CA interacts with its correspondents in the system through its collaboration agent and dedicated agents (communication, coordination and production). The major advantage of our model is the use of the collaboration agent that has a double role: On one hand, it insures a direct link between the various agents in the system, and on the other hand, it constitutes an interface between the user and the CA. The system is technically presented as a multi-agent server that intercepts various users' interactions in order to process them (connection, communication, coordination, production, etc.). A CA is created by the class *GenericAgent* for each client (Figure 8).

This groupware consists of making common work and task coordination feasible between distributed users working on the same teleoperation mission. Figure 9 illustrates the transformation of the single-user ARITI into a collaborative multi-user system, ARITI-C. Hence, every user connected to ARITI-C will collaborate with other connected users, independently of their geographic locations, in order

Figure 8. An example of two collaborator agents generated by two connected clients

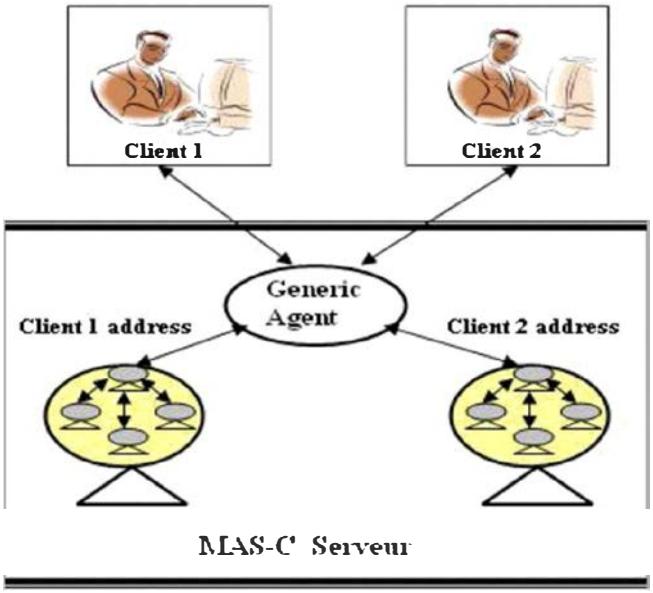
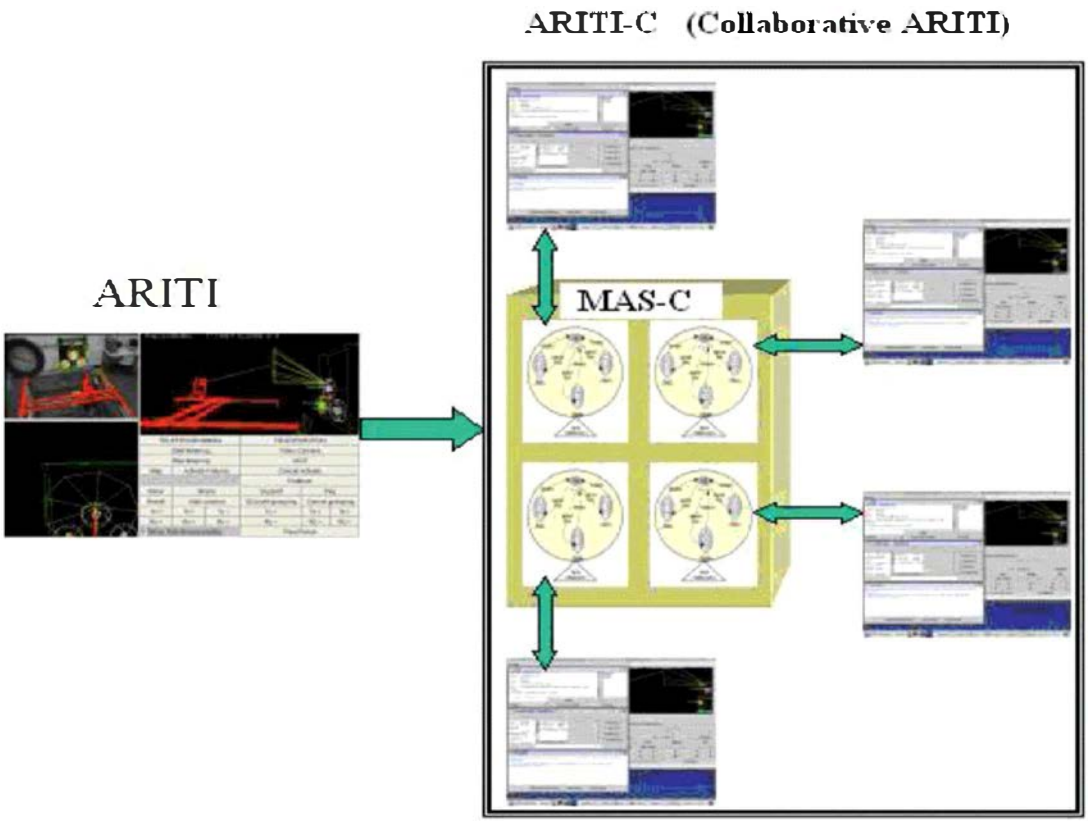


Figure 9. Transformation of the ARITI system to the ARITI-C



to control the real robot. The user have access to many functionalities, such as a textual chat, supervision of users' group, manipulation of lights on the real robot (turn on/off), etc.

Global Software Architecture

In the ARITI-C system, a generator agent creates the Collaborator agents and handles the global functioning of the system: Event logging, identity verification, etc. As we only have one client to command the robot at one particular instant, it is for the Generator agent to centralize and pass the commands to various users. Finally, a Database Management System (DBMS) handles the information resulting from the collaboration process. In fact, ARITI-C users are network clients for:

- The generator agent, which will identify these users and create the Collaborator agents when there is need to do so. Recall that there is one Collaborator agent for every client connected;
- The communication agent, which handles the discussions in a group and serves in creating a mission. In fact, we consider the creation of missions as part of the communication, because it is solely the master user that creates it, which is transmitted to other users. Thus, during the choice of the missions, there is no coordination between users;
- The coordination agent, which redistributes the actions for a chosen mission. Also, this agent is responsible for synchronizing various clients in the production phase;
- The production agent, which handles the use of the real robot and the supervision of missions. This agent notifies the client for the start of the production phase, as well as robot's positions;
- The light and video servers.

The resulting software architecture is described in Figure 10. For visibility reasons, the other agents (collaboration agents that do not communicate directly with the client) are not represented. In Figure 11, we present the software architecture represented by a sequence diagram.

Collaboration between Entities

In this section, we present the protocol used to communicate between the clients and the

Figure 10. Simplified client-server architecture for the ARITI-C system

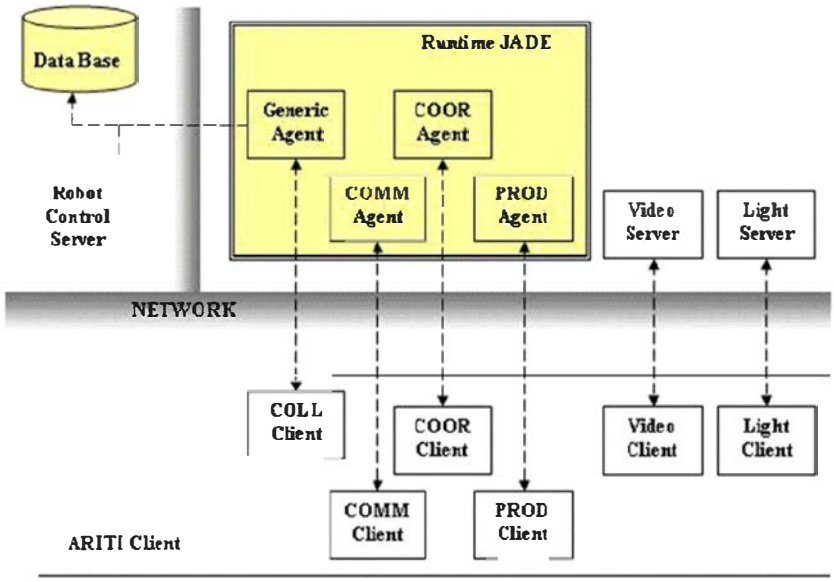
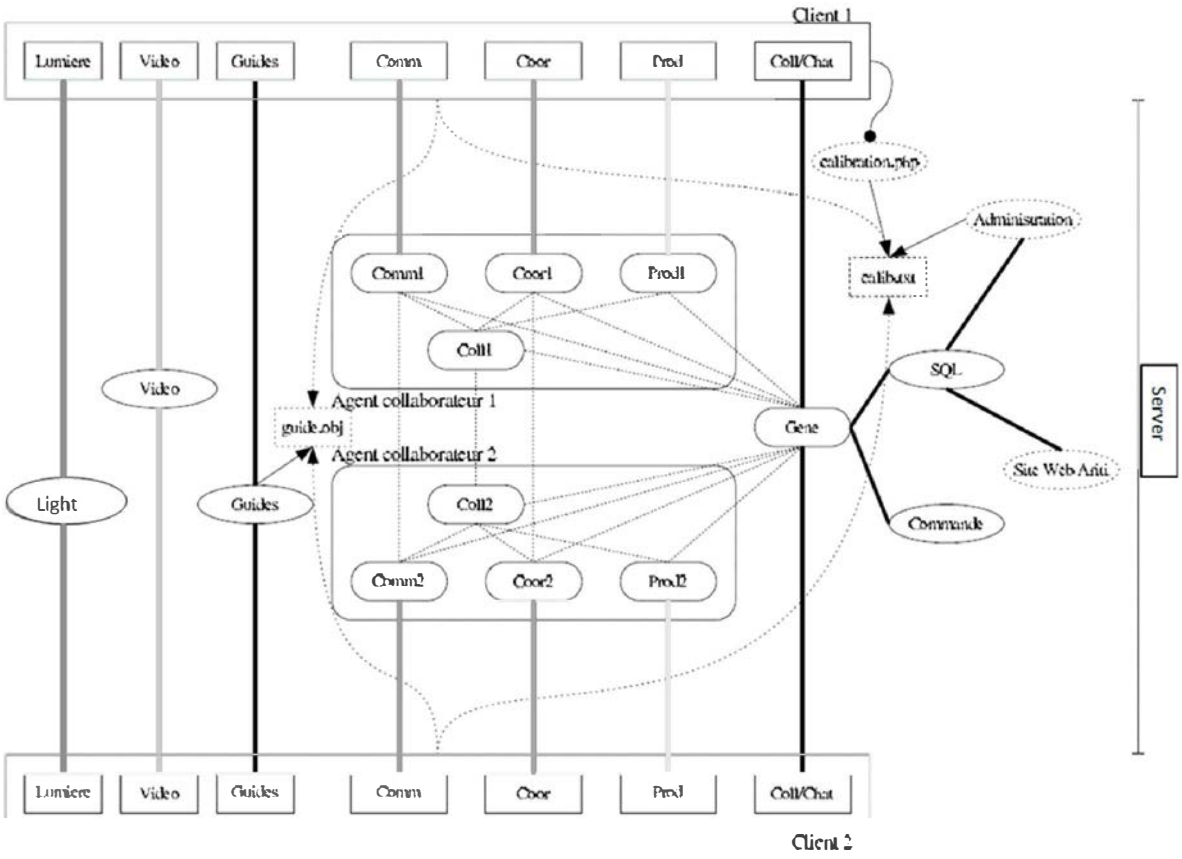


Figure 11. Global architecture of the system represented by a sequence diagram



agents, as well among agents themselves. We describe the various protocols used that enable clients to pass from one collaboration phase to another. Note that the Collaborator agent model has been modified in order to adapt to teleoperation's requirements. Hence, it is not possible to automate the missions' and actions' choices (as described in section 4), as it is for users to decide (Hence, *ContractNet* and *RationalEffect* protocols are not used). Similarly, as only one group can manipulate the robot at each time, we chose for simplicity purposes to only allow the creation of one group at a time, which simplifies system administration. To compose a group, a class called *ClientColl* maintains all users' information. The first connected user is the master user, managing group creation and the validation of the missions and actions, which also has a specific interface for group composition. Hence, when the system receives group's composition by the master user, the communication phase starts: a message is sent to all users connected, and the groups' information is saved in the database:

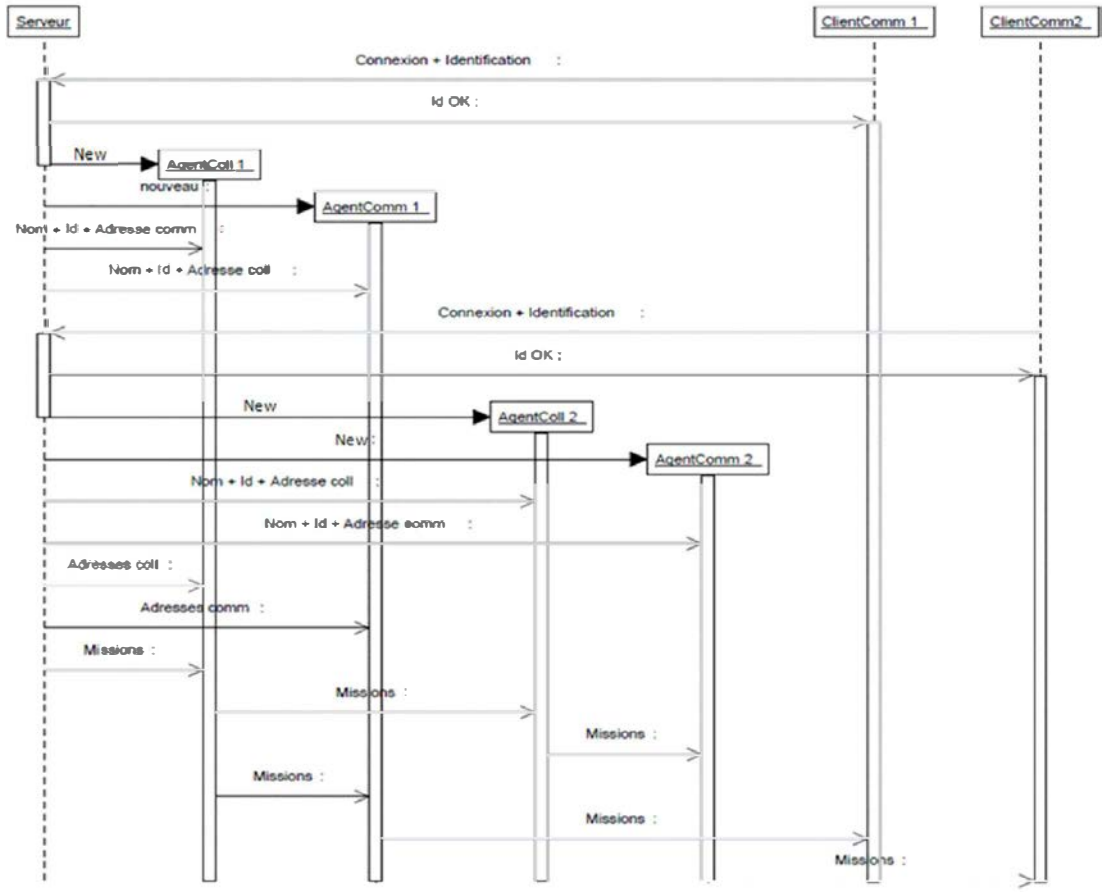
- **Communication phase:** To enable communication, the server must wait for all clients to connect. When a client connects, a collaborative agentand a communication agent are created, which will receive users' information such as name, id, as well as the agents' address that is in charge of this particular user. Once the entire group is created, the collaboration agent of the master user receives the addresses of other collaboration agents. The same thing happens with the communication agent of the master user (receiving the addresses of other communication agents

in the system). Finally, the list of missions is sent the master collaboration agent that relays it to clients. The sequence diagram of the communication phase is shown in Figure 12.

This diagram presents the various messages sent during the communication phase involving two clients of the group. Note that Client 1 in our case is the master client. At the end of this process, agents can communicate with each other, each having the missions' list, as well as a list of the clients in the group. Once communication is initiated, a chat mechanism allows the choice of missions to transit through the communication agents of the group, while each agent relaying the information received from its client to other agents. However, the choice of the mission can only be done by the master user. The chosen mission is returned to the server and saved in the database:

- **Coordination phase:** The initialization of this step is similar to the communication phase. When a client connects, a coordination agent is created and sends its address to the collaboration agent. Once created, the coordination agent is notified by the server for being the "master" agent. Once all the agents expected in the group are identified in the server, the list of missions is sent to each of them. Finally, the master (coordination) agent receives the list of its agents, which enable it to send the list of actions to the client, as well as to send its own address to other coordination agents.

Figure 12. Sequence diagram for the communication phase



Once these agents receive the address of the master agent, the coordination phase can start by sending the list of missions to their corresponding agents. In the diagram shown in Figure 13, a group consists of two clients, with the first one being the master user. After the creation of all agents, the master agent receives the list of its agents, which enables the start of the coordination phase. This process is visible to the client by receiving the list of actions.

During the stage of actions' distribution among clients, the coordination agent can choose between two scenarios: If it is not the master, it relays the actions chosen by its client to the master agent, which determines if this action can be associated with the client. All agents are then notified of the status of the current selection process. However, if it is the master client that chooses an action, it is directly handled by his/her agent, while the status of missions is sent to all agents.

When the master client validates the actions, the production phase is launched. Initially, each coordination agent receives the actions associated with its production agent. This information is sent to the corresponding collaboration agent. Hence, when receiving a message in order to execute an action x, the collaboration agent determines if this action has to be performed by the production agent. Finally, after these messages are transmitted, the message notifying the shift to the production phase is sent to the clients. An example of this scenario is presented in Figure 14. The clients exchange their choice of missions through the master coordination agent. Once the master agent decides to validate the actions, they are distributed to each agent for the smooth shift to the production stage:

- **Production phase:** The same process is used for the creation of production agents: When clients connect to the system, and when all agents are created and their ad-

Figure 13. Sequence diagram for the choice of actions

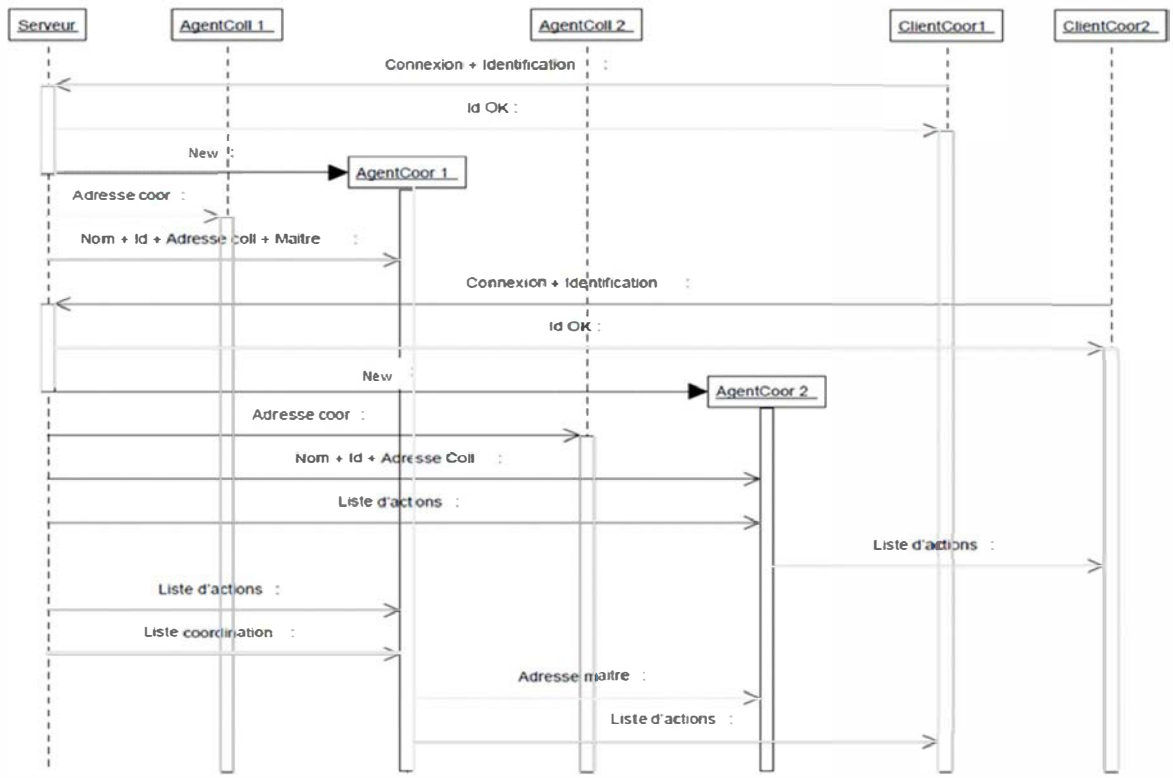
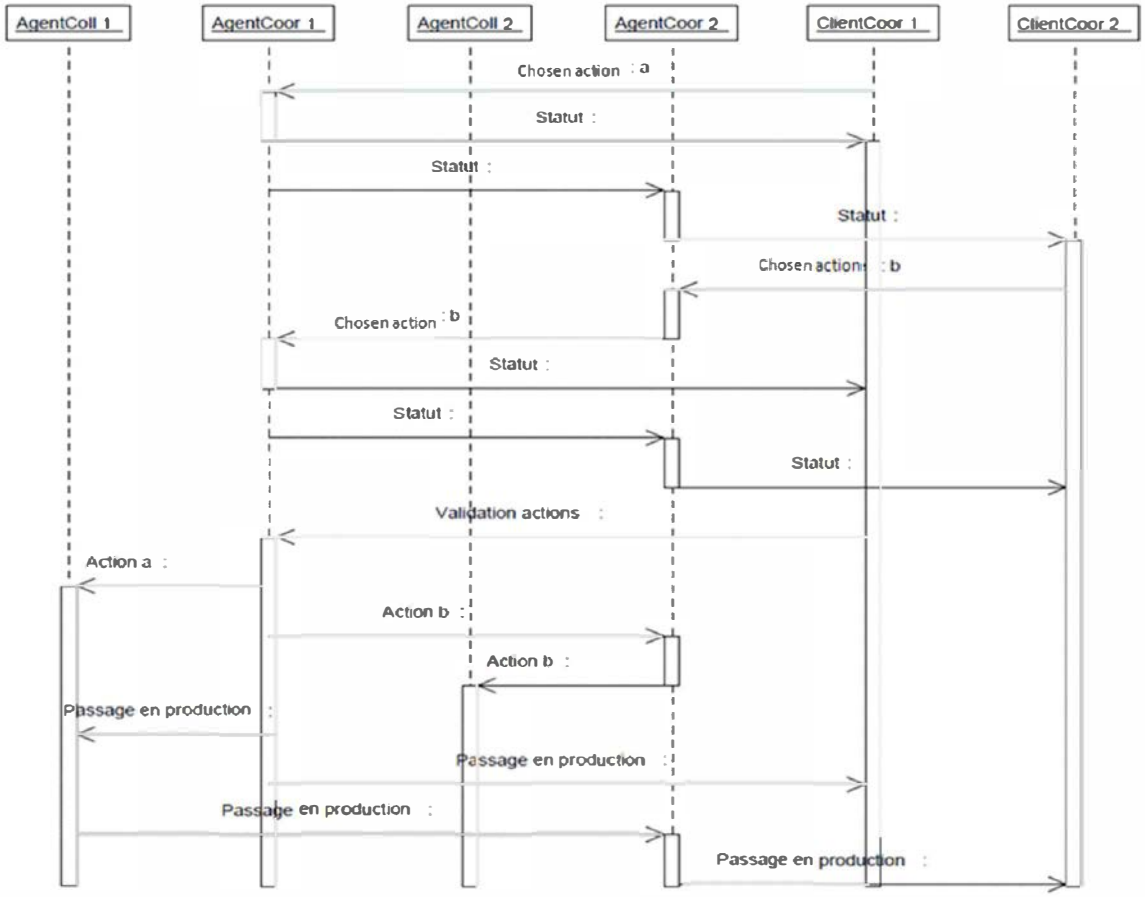


Figure 14. Sequence diagram for the validation of missions



dresses are identified, a signal is sent to the master coordination agent, which sends in return the code of the first action to its collaboration agent. The later relays the information to all other collaboration agents of the group. When a collaboration agent is notified of the first action to execute, it insures that this particular action exist in its list of actions. If it is the case, the agent sends a message to its production agents, prompting the switch to production. Otherwise, this agent will receive a message concerning supervising the action (rather than executing it). Once the action has started, the client concerned with this action sends periodically the status of production to its production agent. This information concerns the robot's position as well as the state of the manipulated cylinders (this process is done automatically). The information is relayed to all production agents of the group through their collaboration agents. Once the production agent receives a message from its client concerning the end of production, the information is transmitted to all coordination agents via the collaboration agents. At this stage, the master coordination agent launches the next action. If none is left, it sends an end of mission signal to the server, which reinitializes the clients' status, destroys the agents and deletes the group for a new cycle to begin.

Analysis and Evaluation for Collaboration

We have developed a tool for the analysis and evaluation of the collaboration process in our system based on a Web interface and a MySQL

database. This interface also allows the administration of different data used or generated by the MAS. As for example, information concerning users, groups, connections, missions, actions, duration of every collaboration phase, etc. The PHP interface gives various features for database administration, as well as for collaboration's analysis: table modifications, sending emails to users, modifying the missions and actions, regenerating a camera's calibration file, exporting and restoring the database and extracting the statistics by user and by group:

- **Evaluation objectives:** The PHP and the database in ARITI-C monitor the system's correct functioning, as well as extract characteristics related to tasks and users: influence of mission parameters on users' performance, influence of the learning process for a single user or a group, influence of group's composition, etc. This fact allows to evaluate how the collaboration is being built for complex tasks, as well as to assess the interest of using the system for other types of missions. The main features of our evaluation are:
  - The relative importance of the various collaboration phases. Hence, the duration of each phase is measured, as well as the number of messages exchanged;
  - The way these collaboration phases (communication, coordination and cooperation/production) vary in significance depending on users' expertise. Ultimately, it is assumed that most of the time is spent in the production phase, after users have accustomed to the missions. One can imagine that eventually the system will offer users an ideal distribution of tasks according to their expertise;
  - The difficulty of a mission and its complexity in a quantified manner. We assume that similar missions will have a similar profile in terms of duration, space distribution, etc.;
  - The influence of ergonomics on the performance. While we follow the evolution of statistics, one can determine the correct or malfunction of certain aspects, or even test alternative ones. Hence, the aim is to make the handling of the software optimal.

The extracted data are:

- The time of every collaboration phase (communication, coordination and production) once every mission is done, which allows calculating the average time as well as the standard deviation;
- The number of messages exchanged in a group. This is done chronologically and by group, which enables to follow group's evolution, or by mission, which enables to assess its quality.

In the following, we present the analysis of ARITI-C's usage statistics:

- **Interpretation of usage statistics:** To facilitate interpretation of results, we first present the protocol used:
  - Groups are made up of three users who have never used the system before. Six groups are created;
  - These groups have repeated the same missions at least 4 times. These missions are constituted of 4 virtual actions (manipulation of only the virtual robot);
  - The experimental conditions are as realistic as possible, with only a textual communication and minimal instructions given.

We observe that, despite differences in duration between groups, the behaviors are essentially the same: the first manipulation is the longest and involving a larger amount of messages as well as an increased communication phase. However, once the users get a grip

of the missions that follows, the focus is shift to the coordination and production phases. We realized that the learning process is quite fast, which is shown in Figure 15. By classifying histogram bars in the order of trials within the group, we see that there is a clear downward trend over the trials, resulting in an increase in coordination and production (see the graph at the right). For readability reasons, the curve representing the number of messages used is not displayed. Tests are classified in the following order: the first six bars represent the first trials of each of the six groups, while the six other bars represent the other trials. This helps to better bring out the learning factor.

In the second observation (see Tables 1 and 2 as well as Figure 16), we find that the distribution of communication phases in terms of duration is similar in the trials of all groups: around 10% of communication time, 25% for coordination and 65% for production. The communication phase has the greatest proportion of standard deviation, as it essentially varies during the learning process. In our third observation, and after enhancing the ergonomics of users' interfaces, we observe a decrease in the overall duration of the manipulation. This fact suggests that modifying the interfaces has been an improvement. In fact, the extraction of statistics allows to only to evaluate the group, but also to improve software quality.

In practice, the modified part of the interface concerns solely the communication and coordination aspects: we clearly realize this fact on the left graph shown in Figure 16: the coordination is faster and the communication is lower as there is less ambiguity in the interface (after the first 10 histogram bars). For clarity

Figure 15. Representation of the learning process for the chosen mission

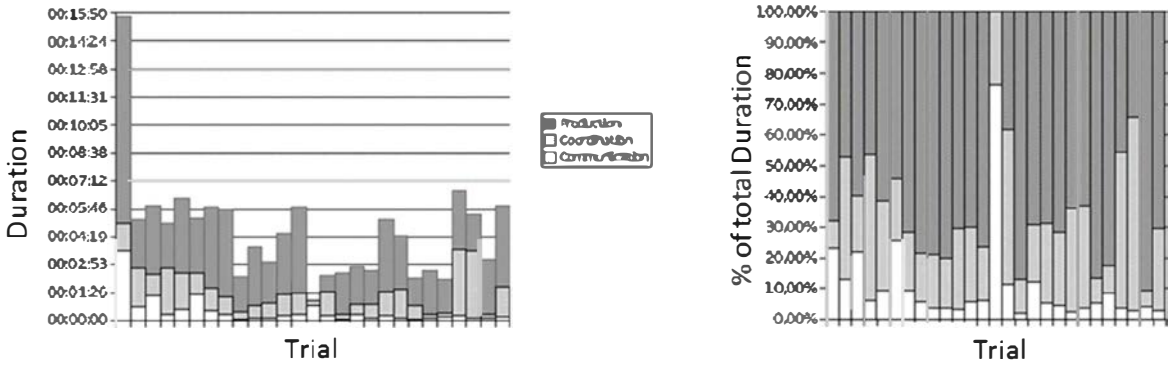


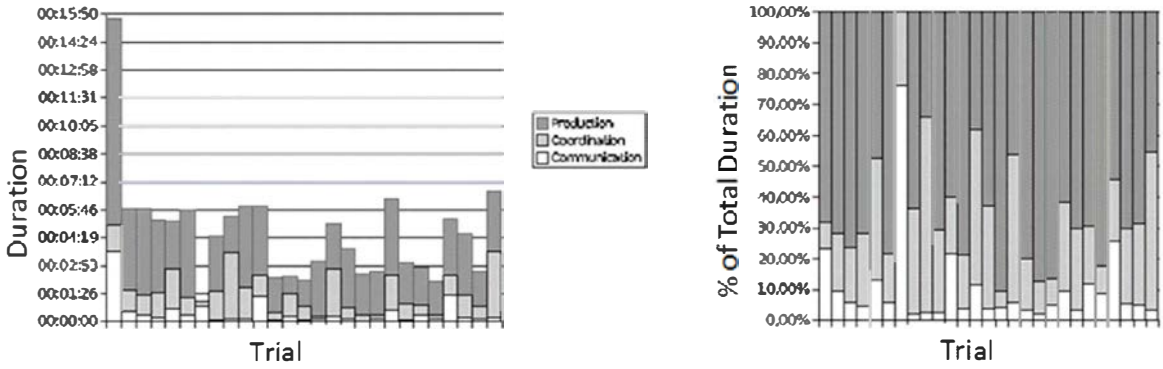
Table 1. Summary of extracted data for all manipulations concerning a particular mission

Date	Duration	Message/Person	Duration Comm	% Comm
Average	4:42	4:04	0:27	9.7
Standard Deviation	2:39	2:96	0:42	-

Table 2. The rest of the extracted data for all manipulations concerning a particular mission

Date	Duration	% Coor	Duration Prod	%Prod
Average	1:11	25.24	3:3	65.05
Standard Deviation	0:51	-	1:47	-

Figure 16. Representation of the mission evolution



reasons, the curve representing the numbers of messages used is not displayed. The tests are classified by chronological order to better perceive the influence of interface's evolutions on mission's accomplishment:

- **Interpretation of results - Drag/drop missions:** In this section, we present the results of overall manipulations for drag/drop missions. In total, 42 trials have been made using both the virtual and the real robot. In fact, the drag/drop missions concern moving objects from one particular place to another. Hence, the users will have 42 trials in manipulating the virtual robot. Every manipulation of the virtual robot induces a manipulation of the real one: the real robot will apply the manipulations of the virtual one; hence, the real robot will also be doing 42 drag/drop missions.

The extracted statistics are compiled in Tables 3 and 4 and shown in Figure 17. These statistics slightly affect the distribution 10%/25% /65% of the observed durations of the collaboration phases: in a real situation, the proportions seem to be 10%/30%/60%. If the missions requiring the largest amount of actions are the longest to execute, this is at least partially offset by the duration of the coordination phase having the same duration. We also observe that there are some aborted missions, which do not significantly influence the observed results under the "ideal" conditions of the trials performed.

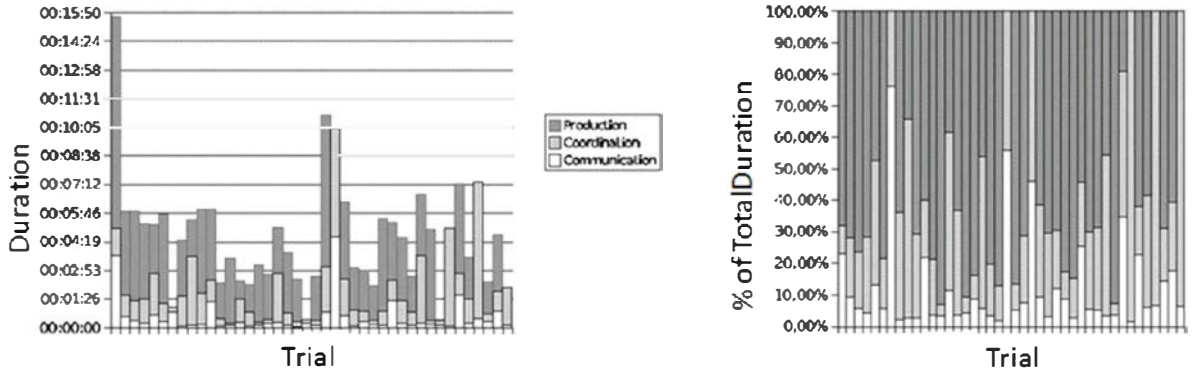
Table 3. Summary of the extracted data for all manipulations concerning a drag/drop missions

Date	Duration	Message/Person	Duration Comm	% Comm
Average	4:36	3:44	0:32	11:82
Standard Deviation	2:48	4.5	0:52	-

Table 4. The rest of the extracted data for all manipulations concerning a drag/drop missions

Date	Duration Coor	% Coor	Duration Prod	% Prod
Average	1:22	29.85	2:41	58.33
Standard Deviation	1:27	-	2:2	-

Figure 17. Representation of all results for drag/drop missions



To sum up, we have extracted statistical data in order to analyze and evaluate the collaboration process in our system. The results are promising, and clearly show that all agents in the system work proportionally according to the task being executed.

CONCLUSION

In this paper, we presented a conceptual model, the Collaborator agent, intended to design collaborative software architectures based on multi-agent systems. The proposed model is defined as a set of four agents (collaboration, communication, coordination and production) supporting the whole set of collaborative tasks. We discussed implementation and evaluation issues of the Collaborator agent. Our research work is resumed by the presentation of the ARITI-C system as a collaborative teleoperation Interface via Internet. The design of this system is based on research done in both CSCW and MAS domains. The PHP interface and the database of the ARITI-C system were conceived for tracking the functioning of the system, as well as extracting relative characteristics related to missions and users. These components allow to evaluate the way the collaboration process is constructed for complex tasks, as well as to justify the use of our system in other application domains. However, the Collaborator agent has still a static structure. In other words, modifications in its software architecture is yet not as easy, as the system will have to be inactive and shut down in order to integrate new modules/missions, recompile the code and start all over for the modifications to take effect. In fact, users' needs are increasingly growing in order to fit the collaborative application into their preferences, and not the contrary. For Fuks, Raposo, Gerosa & Lucena (2005), the designer of groupware applications must bear in mind that collaborative applications should be flexible enough to adapt to group characteristics and to the evolution of work processes. One property seems emerging in the design of groupware: tailorability. Hence, our aim on the long run is to integrate the concept of tailorability to design a more flexible groupware system. In fact, we aim at using Web services, which are nowadays widely used in the industry, along with ontologies for bringing semantic definitions to software objects. Therefore, we want the behavior of the Collaborator agent to dynamically adapt to users' needs and preferences. This behavior should be created without hard coding and reexecuting the application, which consume time and eventually necessitate the intervention of an expert programmer instead of a normal end-user. We believe that these concepts will bring an innovative approach to design tailorable collaborative applications based on MAS, and hence shifting the multi-agent concept from theoretical simulation to real practice in the industrial world.

ACKNOWLEDGMENT

This research was supported by the Digital Ocean project funded by the European Commission (FP7 program).

REFERENCES

Arlabrosse, F., Gleizes, M. P., & Occello, M. (2004). Méthodes de conception. In Systèmes Multi-Agents. Paris, France: OFTA editors.

Cabri, G., Ferrari, L., & Leonardi, L. (2004). Towards the use of mobile agent based message systems. In *Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 04)* (pp. 27-32).

Cabri, G., Ferrari, L., & Leonardi, L. A. (2003). Case study in role-based agent interactions. In *Proceedings of the 12th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 03).*

Coleman, D., & Shapiro, R. (1992). *Defining groupware.* Special Advertising Section to Network World.

Cragg, L., & Hu, H. (2003). Application of mobile agents to robust teleoperation of internet robots in nuclear decommissioning. In *Proceedings of the IEEE International Conference on Industrial Technology,* (Vol. 2, pp. 1214-1219).

D'Inverno, M., Fisher, M. A., Lomuscio, M., Luck, M., de Rijke, M. R., & Wooldridge, M. (1997). Formalisms for multi-agent systems. *The Knowledge Engineering Review*, *12*(3). doi:10.1017/S0269888997003068.

D'Inverno, M., Kinny, D., Luck, M., & Woolridge, M. (1998). A formal specification of dmars. In Wooldrige Singh, Rao (Eds.), *Proceedings of ATAL98 Workshop on Agent Theories, Architectures, and Languages,* (Vol. *LINAI 1365*, pp. 155-176). Amsterdam, the Netherlands, Springer-Verlag.

Ellis, C. (1999). Workflow technology. In M. Beaudouin-Lafon (Ed.), *Computer supported cooperative work of Trends in Software* (Vol. 7, pp. 29–54). John Wiley and Sons.

Ellis, C. A., Gibbs, S. J., & Rein, G. (1991). Groupware: Some issues and experiences. *Journal of communications of the ACM (CACM), 34*(1), 38-58. ACM Press.

Ellis, C. A., & Wainer, J. A. (1994). Conceptual model of groupware. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW '94)* (pp. 79-88).

Ferber, J. (1997). Les systemes multi-agents: Un Aperçu General. *Revue of Techniques et Sciences Informatiques*, *16*(8).

Ferber, J., & Muller, J. P. (1996). Influences and reaction: A model of situated multi-agent systems. In *Proceedings of the ICMAS96*, Kyoto, Japan.

Frey, D., Stockheim, T., Woelk, P., & Zimmermann, R. (2003). Integrated multi-agent- based supply chain management. In *Proceedings of the 12th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 03).*

Fuks, H., Raposo, A. B., Gerosa, M. A., & Lucena, C. J. P. (2005). Applying the 3C model to groupware development. *International Journal of Cooperative Information Systems*, *14*(2), 299–328. doi:10.1142/S0218843005001171.

Fuks, H., Raposo, A. B., Gerosa, M. A., Pimentel, M., & de Lucena, C. J. P. (2007). The 3C collaboration model. Journal of the Encyclopedia of E-Collaboration, Ned Kock (org), 637-644.

Glaser, N. (2002). Conceptual modeling of multi-agent systems: The Comomas engineering environment (multi-agent systems). In *Artificial societies, and simulated organizations.* Kluwer Academic Publishers.

Khezami, N., Otmane, S., & Mallem, M. (2005). A new interface for collaborative teleoperation. *International Federation of Automatic Control (PRAHA IFAC World Congress).*

Laurillau, Y., & Laurence, N. (2002). Clover architecture for groupware. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW'2002)*, New Orleans, LA.

Lewis, B., Tastan, B., & Sukthankar, G. (2010). Improving multi-robot teleoperation by inferring operator distraction. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems* (Vol. 1, pp. 1505-1506).

Lin, C. H., Song, K. T., & Anderson, G. T. (2005). Agent-based robot control design for multi-robot cooperation. In *Proceedings of the IEEE international conference Systems, Man and Cybernetics,* (vol. 1, pp. 545-547).

Moulin, B., & Brassard, M. (1996). A scenario-based design method and an environment for the development of multi-agent systems. In *Proceedings of the First Australian Workshop on Distributed Artificial Intelligence (LNAI volume 1087)*, 216-231. Heidelberg, Germany: Springer-Verlag.

Moulin, B., & Cloutier, L. (1994). Collaborative work based on multi-agent architectures: A methodological perspective. In *Proceedings of the Soft Computing: Fuzzy Logic, Neural Networks and Distributed Artificial Intelligence* (pp. 261-296). Prentice-Hall.

Occello, M., Koring, J., Ibriz, A., & Erradi, M. (2002). A multi-formalism method for designing and validating intelligent agent-based systems. In *the Proceedings of Intelligent Processing Conference*, Beijing, China (pp. 53-58). IEEE Computer Society.

Oliveira, F. F., Antunes, J. C. P., & Guizzardi, R. S. S. (2007). Towards a collaboration ontology. In *Proc. of the Snd Brazilian Workshop on Ontologies and Metamodels for Software and Data Engineering.*

Otmane, S., Mallem, M., Kheddar, A., & Chavand, F. (2000). Active virtual guide as an apparatus for augmented reality based telemanipulation system on the Internet. *In IEEE Computer Society. ANSS, 2000*, 185–191.

Picard, G., & Gleizes, M. P. (2004). The Adelfe methodology. In *Methodologies and Software Engineering for Agent Systems* (pp. 157–175). Springer. doi:10.1007/1-4020-8058-1_11.

Ricordel, P.-M., & Demazeau, Y. (2002). VOLCANO: A vowels-oriented multi-agent platform. In *Proceedings of the International Conference of Central Eastern Europe on Multi-Agent Systems (CEEMAS`01), From Theory to Practice in Multi-Agent Systems, Dunin-Keplicz and Nawarecki, eds, LNAI 2296* (pp. 252-262). Springer Verlag.

Russell, S. J., & Norvig, P. (1995). *Artificial intelligence: A modern approach.* Prentice Hall.

Selliah, S., Reddy, R., Yu, J., Bharadwaj, V., & Reddy, S. (2004). Eksarva: A framework for enabling agent-based collaboration. In *Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET-ICE 04)* (pp. 33-38).

Wegner, R., & Anderson, J. (2006). Agent-based support for balancing teleoperation and autonomy in urban search and rescue. *International Journal of Robotics and Automation*, *21*(2), 120–128. doi:10.2316/Journal.206.2006.2.206-2796.

Wooldridge, M., & Jennings, N. R. (2005). Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, *10*(2), 115152.