22

Formal Design of Cooperative Systems

Michel DIAZ, Thierry VILLEMUR and François VERNADAT
LAAS du CNRS
7, Avenue du Colonel Roche
31077 TOULOUSE Cedex - FRANCE
e-mail: {diaz, villemur, francois}@laas.fr

Abstract

This paper starts with a formal definition of the concept of cooperation in distributed systems. The proposed model is based on the use of graphs and of logic, where logic expresses contradiction and pragmatism. The dynamic structuring of such systems leads to the concept of multi-connection at the cooperation level and to a layered top-down formal design methodology. The protocol providing the multi-connection management or membership service will be specified and verified using the VAL tool and a Petri net based specification. It will be shown how the resulting system can be proven correct. The VAL description is then translated into an Estelle description that has been implemented on top of a distributed platform. Private possible conversation subgroups (PCSs) inside a cooperative group of agents have also been introduced.

Keywords

Cooperative work, cooperative groups, protocol design, multi-agent system VAL, Estelle, formal specifications

1 INTRODUCTION

Several articles have introduced CSCW [KRAE88], [BANN89], [ELLI91], [KRAS91], [BLAI94]. Computer Supported Cooperative Work (CSCW) in its broader sense includes human sciences, sociology, cognitive sciences, distributed artificial intelligence, distributed architectures and communication networks [BANN89]. "CSCW applications are computer-based systems that support groups of people engaged in a common task and that provide an interface to a shared environment" [ELLI91]. The software part of the CSCW, the groupware [KARS94], develops new software tools, platforms and applications, for groups of agents to work in common.

Due to their complexity, cooperative groups and cooperative entities must use models to describe their interactions and behaviours: Petri net-based models have been used for

P. Dembiński et al. (eds.), *Protocol Specification, Testing and Verification XV*

synchronising cooperative office information systems, e.g. for detecting conflicts that happen when the tasks are processed [VICT89]. Petri net extensions are also used for describing the workflow management of office information systems [AGOS94], [AALS94]. The behaviour of cooperative agents can be represented with the actor model. In large systems, the actors can be classified into groups called organizations of restricted generality [HEWI91] that provides common resources to all its actors. Object oriented models are also used to model groups handling information in an asynchronous way. In [BENF93], groups are called application domains that contain data and people. All the entities contained inside a group (items, people, domains...) are described with classes that inherit from the group communication class of a particular group.

Cooperation and groups of agents

Nevertheless, up to now, much less attention has been paid to develop formal models to understand the basic concepts that imply the selected groups or domains. Defining such a high formal level is the purpose of this talk. It considers the cooperative framework in which the distributed behaviour occurs and furthermore will be based on a set of formal models. How these models express high level cooperation and how they appear in extended cooperation architectures will be discussed in the sequel.

In order to design cooperative systems, many aspects are needed, starting from the common task and ending with the corresponding implementation. It follows that the designers have to consider a formal definition of cooperation and a methodology for producing the distributed software that supports the cooperative work.

The first needed model, the formal cooperation model [DIAZ92], [DIAZ93a], will represent the information structure and the information relationships between the cooperative agents. It organizes the exchanges of information and the structure of the group of agents. The definition of cooperation is based on private knowledge and information sharing: one agent is in cooperation with another one if it gives to it access to part of its own private knowledge. In order to check non contradictory behaviours, the agents will use a dedicated view of formal cooperation that will be based on logic.

From the previous general model, three extensions are considered:

- dynamic cooperative groups whose structure changes in time are introduced. For each cooperative group, a set of configurations, called valid instantaneous configurations, gives the subsets of agents that are semantically significant at any instant of the cooperative work. The group dynamicity results from the requests for entering or leaving the cooperative group.
- groups can be composed of several cooperation domains. Complex cooperation is cut in small parts, these parts possibly being independent one from the others in some aspects. This may lead to inconsistencies.
- dynamic creation of private conversation subgroups (PCS) is allowed. The PCS are very dynamic private subgroups created inside cooperation by subsets of cooperative agents to realize parallel interactions between subsets of agents. The PCS structure follows the one given by the cooperation.

Cooperation and communication

We have then emphasized the distinction between communication and cooperation. Communication follows a hierarchical OSI [ZIMM80] Reference model-based architecture, cooperation being defined on top of communication. Also, a distinction is made between

cooperation and decision, because decision is assumed to be located in a higher layer level than cooperation and dependent on cooperations. Then all decisional mechanisms, including entity internal constraints such as beliefs and desires [FAGI85], [SHOH89], are not considered in this paper that follows a four-layered model: communication, cooperation, decision, application.

Presently, the design of communication protocols is based on two specifications: the requested properties and the expected behaviours. These specifications are given using different formal approaches, as CCS, Petri nets, Formal Description Techniques (Estelle, LOTOS,...) [MANN89], [MILN80], [DIAZ82], [EIJK88]. Different logics have been developed to specify system properties, including Modal Logic [HUGH68] for expressing and proving safety and liveness properties. These approaches proved to be of high interest for the design of complex ISO, CCITT and IEEE protocols and services.

Design methodology

This article suggests a design methodology including the following steps:

a)- definition of the information that belong to the cooperative agents and definition of the cooperation, i.e. of the data that is made available by one agent to the other agents it cooperates with. This step defines an interpreted graph, i.e. a graph typed by data, predicates, and flow of information;

b)- definition of the behaviours of each of the agents. This defines the role of the participants and the distribution of the functions and work, leading to the global definition of the cooperative group. The cooperation group is structured and the relations between its members can be represented by a graph;

c)- definition of the cooperation software. In order to design the software architecture, a set of services and protocols are defined on top of a communication system, i.e. on top of the Transport layer or of a higher layer. The software is specified, validated and implemented by using formal approaches;

c-1) As the structure of the cooperative group may dynamically change in time according to the entries and exits of users, this dynamic behaviour will in general be very complex, and it has then to be verified. A formal model based on VAL, a Petri net based description of dynamic agents, has been used. The behaviours, including the structural changes of the group, are produced and a validation procedure is conducted;

c-2) The VAL validated behaviours will then be translated into an implementation model. Of course this model must not be far from the validation model and from the generated code. Estelle has been chosen for the translation to be automatically compiled.

Such a methodology has been followed for designing a basic support software for cooperative systems. It contains the graph model, the handling and membership of the dynamic agents and includes the formation of private cooperation subgroups, where subgroups of agents can form private conversation subgroups (PCSs) to exchange information privately inside these subgroups. This will be given in the next sections.

Contents

Section 2 presents the formal cooperation model. Section 3 presents the design methodology when applied to the design of a dynamic structured group membership algorithm. Section 4 gives how private conversation subgroups have been handled.

2 BASIC MODEL FOR COOPERATION

The model has been developed to formalize an intuitive meaning of cooperation. General structuring and logic values are considered for complex cooperative systems.

2.1 Model for cooperative systems

Cooperation will be defined as what seems to be its weakest sense, where cooperation means giving access to private information (Figure 1): as a consequence, one agent cooperates with one other if the former makes its information available to the latter [DIAZ92], [DIAZ93a].

Let A be a set of agents, $A = \{A_i\}$. Each agent, A_i, maintains a set of information, data and predicates, and cooperates with other agents. Let $P = \{P_i\}$ be the data exported by a given agent A_i and so the data that can potentially be known outside A_i.

Definition: A_i cooperates (is in cooperation) with A_j if A_i allows A_j to know the values of some data exported by A_i; the exported values become known by A_j.

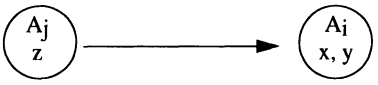

Figure 1 Agent A_i cooperates with agent A_j; A_j can read the values "x" and "y" owned by A_i, but A_i can not read "z"

Many proposals use non classical logics to express the relationships and properties amongst communicating agents [MANN89]. Most of them are based on the possible world semantics of Modal Logic. Furthermore, Modal Logic can easily be related to the set of all possible distributed computations and so allows one to check global behaviours [EMER89]. Cooperations will consequently be defined as sets of different semantics, embedded in adequate logical systems.

Modal Logic is the logic of possibility and necessity. For Modal Logic, validity is based upon Kripke structures defined as ordered triples <W, **R**, V>, where:

- W is a set of objects or worlds $W = \{w_i\}$
- **R** is dyadic relation defined over the members of W and
- V is a value assignment.

Let **R** be a relation on the members of A such that, if one agent, say A_i, cooperates with another agent, say A_j, then the ordered pair (A_i, A_j) belong to **R**: an arc is drawn from A_i to A_j, which means that A_j can know the exported information located in A_i. The graph given in Figure 1 means that A_j can access the predicates exported by A_i. In the general case, given a set A, the cooperation structure among the members of A gives **R**, a directed graph, the cooperation graph, having as many vertices as there are elements in A. Figure 2 gives an example of such a graph that defines the cooperative type and its flow of information.

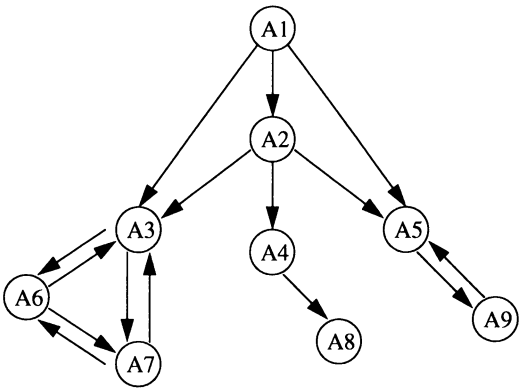

Figure 2 A cooperation graph

Note that such a model gives a meaning different from the classical view of knowledge [HALP86]. The difficulty, when modelling knowledge, is how to select the set of worlds which are accessible by a given agent. When considering usual distributed behaviours, the set of accessible worlds can be nicely defined by the reachability graph, the graph of all possible states [EMER89] [JUAN88]. In the cooperative information graph proposed here, the accessible worlds are no more related to behaviours and have the following intuitive basis: each of the worlds is one agent and the set of worlds that are accessible by agent A_i is the set of agents that are in cooperation with A_i. Consequently, a Kripke structure can be defined that is exactly the cooperation graph, a high level modelling of the real cooperation. Note that this definition only considers information states and not intentional states [WERN88] and the latter are considered to belong to a different conceptual layer.

2.2 Evaluating data and cooperation predicates

Let us now give the semantics for such graphs. In the general case, agents are able to get new values that update their predicates. As a consequence, predicate values may change. Two resulting cooperative behaviours appear to be possible:

- when a information value changes, the new value is sent;
- no information is sent until explicitly requested by a cooperative node.

These two cases will lead to different communication requirements and are no more developed here.

Cooperation Predicates

Let us make the assumption that any information made available to a given agent becomes local to this agent. As a consequence, any information produced by or any predicate evaluated by a given agent A_i is only accessible to the agents linked by the cooperation relation to A_i.

In each node of a model, data are exported or local: it can be exported to the agents that are in cooperation with this node; if exported, it becomes local to the importing node (and not known by the other nodes).

Let us assume now that a predicate has to be evaluated and needs information from a cooperating agent for this evaluation. It follows that there are cases where the needed information may be not available, for instance due to a too old value. As a consequence, a logic predicate P in such a system has to take its value inside a three valued logic system:

1=True, 0=False, and ø, defined as the value of P in all nodes where P is not defined.

It appears [DIAZ92] that ø can have different meanings by the OR connective:

- ø means don't care, and $\alpha \vee ø = \alpha$.
- ø means not computable: $ø \vee ø = ø$ et $0 \vee 1 = 1$, $\alpha \vee ø = ø$ (0 et 1 are needed to compute $\vee$);
- ø means an optimistic view, and $0 \vee ø = ø$, $1 \vee ø = 1$;
- ø means a pessimistic view, and $0 \vee ø = 0$, $1 \vee ø = ø$.

2.3 Characterizing R to define cooperation domains

R is reflexive, and $A_i \mathbf{R} A_i$, because all agents know their own values. Nevertheless, in the general case, **R** is neither symmetric nor transitive. If **R** is symmetric, the cooperation has been called bilateral, if **R** is transitive, (completely) hierarchical and if **R** is both symmetric and transitive, total. The semantics of the cooperation, which define the behaviour of a given agent, will then depend on relation **R**, i. e. the cooperation framework.

Let us now consider Figure 2. Its nodes, which define W and **R,** can be considered different ways. For instance, let us consider first only nodes A3, A6 and A7 and arcs connecting them. Those nodes are in total cooperation, as the cooperation is restricted to vertices A3, A6 and A7 and to the 6 arcs relating them: **R1**, such a restriction of **R**, is symmetric and transitive. On the other hand, restricting now the graph to the set {A1, A2, A3, A5} and to the arcs relating them, leads to **R2**(A1, A2, A3, A5), which shows that the subgraph including A1, A2, A3, A5 is hierarchical. It then clearly appears that A3, for instance, can be considered as being involved in two different (sub)cooperations, one being total, **R1**(A3, A6, A7), and the other hierarchical, **R2**(A1, A2, A3, A5). Also, of course, A3 can be seen as involved in one global cooperation relation, the one defined by all nodes and all arcs existing in Figure 2.

Consequently, A3 can be related to other agents in different settings: selecting {A3, A6, A7} defines a total cooperation for A3; selecting {A1, A2, A3, A5} defines a hierarchical cooperation and selecting the whole graph defines a general cooperation.

Domains of cooperation

Let us now consider the general case where the cooperation graph is very complex, as in Figure 2. In order to extend the model, let us consider that cooperations can be split into different cooperation domains related to coherent sub-problems or to coherent sub-applications. Those domains then define manageable sets of agents.

Definition 2: Let a global cooperation be split into different cooperation domains which form subgraphs of the complete graph. These domains are such that all data in a domain is coherent (is being able to be well understood and managed).

Then, a cooperation domain defines a set of strongly linked agents that have agreed to cooperate and that have agreed on the distribution of relevant data. It follows that predicates cannot evaluate to contradictory values inside one domain. In multi-domains cooperation, each of the predicates can be evaluated by one of the agents of the domain it belongs to.

Let A_j be an agent. A_j may belong to different domains, for instance two, depending on its cooperations. Let us now assume that one predicate P_k is known by A_j and that this predicate appears and has to be evaluated in the two domains A_j belongs to: then, by definition, P_k is consistent with respect to the first domain and is also consistent with respect to the second one, as long as they are considered independently. Nevertheless, and this is the key point, nothing prevents P_k to be incoherent when both domains are considered together, and in particular by A_j, because A_j accesses the two domains (Figure 3): a predicate P_k, can be evaluated as false in one domain and true in the other, leading to a real but coherent contradiction, which does not prevent reasoning if semantics are given.

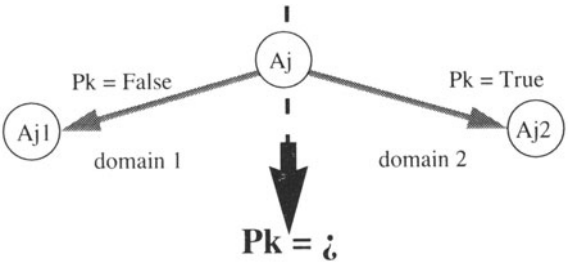

Figure 3 Contradiction between domains 1 and 2

Semantics of contradictions and pragmatics

Consequently, in general cooperation graphs, predicates are assumed to take their values inside a four valued logic:

1, 0, ø, ¿ or Contradiction.

The value assignment has now to be extended to account for this fourth logical value [DIAZ92] and the problem is how to handle cases where contradictions occur.

The previous definitions allow one to define pragmatics in the sequence "syntax, semantics, pragmatics", where pragmatics are above semantics. As a consequence, in [DIAZ92] pragmatic decisions are defined as decisions, above semantics, that select one of the logical value, True or False, i. e. one of the two values, this choice being dependent of the cooperation structure and semantics. A pragmatic decision then means selecting a semantical value instead of another (or others) depending on the structure of the cooperation, i.e. the context in which the different values are evaluated.

3 SERVICES AND PROTOCOLS FOR GROUP MEMBERSHIP

From the preceding general model, three extensions have been proposed: the first one handles dynamic cooperative groups whose structure changes in time. For each cooperative group, a set

of configurations, called valid instantaneous configurations, gives the subsets of agents that are semantically significant to participate to the cooperative work. A communication service and an associated set of service primitives manages the cooperative group dynamicity by taking into account the requests for participating or leaving the cooperative group. A protocol associated to the service, is specified by using the Estelle language and a specific layered architecture, the Open System Interconnection (OSI) model from ISO. The protocol entities are developed on top of a multicast communication service. The protocol has been verified using VAL, a multi-agent environment based on Petri nets, using the analysis of the accessibility graph with projections. The service and protocol are then improved to take into account some disconnection errors coming from the users or from the multicast communication layers and are implanted inside a distributed UNIX environment from the Estelle code generated and translated into C. A second extension has been defined for groups composed of several cooperation domains and some services and protocols have been proposed to manage them. A third extension introduces the dynamic creation of private conversation subgroups (PCS). The PCSs are very dynamic private subgroups created inside cooperation by subsets of cooperative agents. The PCS structure follows those given by the cooperation. PCSs are used to realize between subsets of agents short and brief interactions (in comparison to the cooperation duration). A service and a protocol that follows the OSI model, are specified using the Estelle language. The service creates, closes and manages PCSs and synchronizes them with the dynamic evolution of cooperation.

This chapter presents the dynamic extensions for cooperative groups and the next one details the PCSs.

3.1 Evolution in time

Let us now consider how the activity or the cooperative work associated to the group is initiated. The trivial solution corresponds to the case where cooperation is considered to begin when all agents are ready to start. Of course, this view is too restrictive as cooperative work can be performed as soon as an adequate and sound set of participants comes into existence. This means that at a given instant it is not necessary that all agents are present to begin the cooperative work [DIAZ93b]. Nevertheless, all the participant agents must be considered and a set of communication channels have to be established. This is precisely a multi-connection in the communication sense, but this multi-connection is being defined at the cooperation, logical level, still not considering parameters resulting from the communication level.

As a consequence, the conceptual model has to define which agents must cooperate at any instant of time: the way they interact and their number. Both the structure and the number can change along the cooperative work.

The meaningfulness of the structure is of course dependent on the application associated to the cooperative work. If the previously given graph is considered, then the set of the possible sub-set that can define configurations coherent with the beginning of the work is a set of the subset of the cooperative graph. Among all subgraphs of the graph of cooperation, the application must then being able to choose and pass to the cooperative layer the subgraphs that are semantically possible, i.e. coherent with the start of the requested work. Those subgraphs selected by the application are called valid instantaneous graphs. Figure 4 shows two valid instantaneous graphs that could come from the cooperation graph of Figure 2.

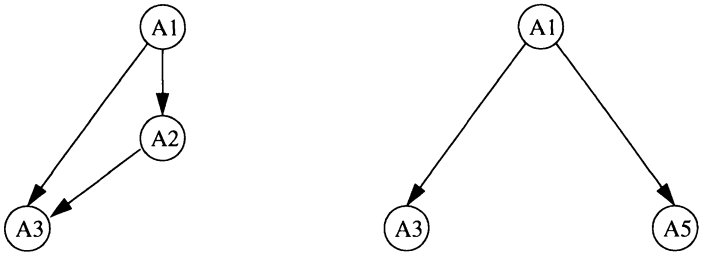

Figure 4 Example of two valid instantaneous graphs coming from Figure 2

Note that the application must be given a simple way to easily make these valid instantaneous graphs known to the cooperation layer: models, rules,... A service and a protocol have been developed for defining and handling the behaviour of all possible valid subgraphs that occur during the life of the cooperative work.

3.2 Role of the service

The proposed service controls the evolution of the cooperation with respect to time [DIAZ93b]. It aim is to pass from a valid configuration where agents are cooperating to another one, by considering the requests of entering and leaving the considered cooperation (Figure 5). Let us consider a set of agents which are cooperating and which form a valid instantaneous graph. Inside the cooperative group, other agents may want to participate to the cooperative work and join those which are already working. Also, agents which are cooperating, may want to leave the work and stop their participation. Considering the requests to enter and the requests to leave the cooperative work, the service tries to form a new valid instantaneous graph.

When the new set of agents is coherent with the valid cooperative subgraphs, the group modification procedure can be started. It has been decided to transmit the set of the requests to the on-going users, i.e. to the present participants of the cooperative work. The agents leaving and entering the cooperation are sent to the software users and the present users are requested to vote to accept or not accept the requesting agents, i.e. to accept or delay the change of the cooperation structure.

Note that when a cooperating agent modifies its own data, it sends the new values to the agents which need that data, i.e. all agents which have arcs linked to it in the graph. This requires multicast mechanisms to transmit new data values between the subgroup agents. It appears that data exchanges can interfere with cooperation changes. As a consequence, data exchanges have to be synchronized with the evolution of the structure of cooperation in time.

Then, when a cooperation change has been accepted, the new cooperating agents must exchange initial information that compose the context of each agent before starting the new phase of the cooperation. It is necessary that they mutually exchange their own contexts according to the cooperation structure; definition of the contexts seems to be application dependent.

The proposed service handles all previous primitives needed for requesting, voting, accepting or delaying the changes of the cooperation graph.

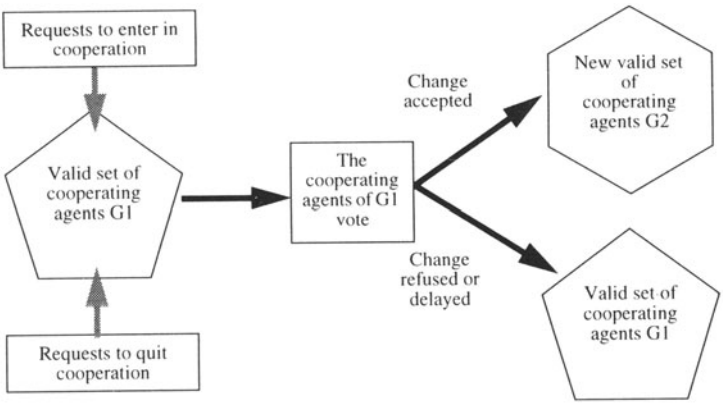

Figure 5 Principle of a cooperation change

3.3 Specification description

The service and its associated protocol have been described using the VAL environment [VERN93]. VAL is a multi-agent programming environment that can be used to conceive, test and verify systems composed of instances of communicating agents. A VAL specification is composed of a set of communicating agents whose behaviour is based on predicates/transitions Petri net. The places and transitions of the Petri net that model the behaviour of each agent are represented using PROLOG predicates. Each agent belongs to a class that gives the generic behaviour and the generic functionalities of all its instantiated agents. Agents can dynamically create other agents in indicating the class of the new created agent. Synchronous and asynchronous communications are possible with multiple message sendings and receivings.

The protocol defined is a semi-centralized protocol [VILL94]. All entry and exit requests are centralized and controlled by a manager. The manager controls and supervises the evolution of the cooperation structure in time. This architecture (Figure 6) requires two classes of agents. As the behaviour of each cooperative agent is the same, a single class can represent them. Each cooperative agent is an instance of this class. In the same way, another class gives the behaviour of the manager. The manager is the unique instance of this second class.

Each cooperative agent sends the requests for joining or leaving the cooperative work to the manager. Also, each agent sends to the cooperation manager its vote when a cooperation change is requested. The agents enforce the synchronisation phases between exchanges of data, and changes of cooperative structure. Before the cooperation structure change, the agents put their own data in a coherent state and they exchange between them the initial data just after a cooperation change. The subgraph validity criteria have been embedded into the cooperation manager. The manager contains also the current state of the cooperative agents, determines whether a new valid instantaneous graph is possible. If it is possible, it manages the voting process to know whether the cooperative agents accept the cooperation change.

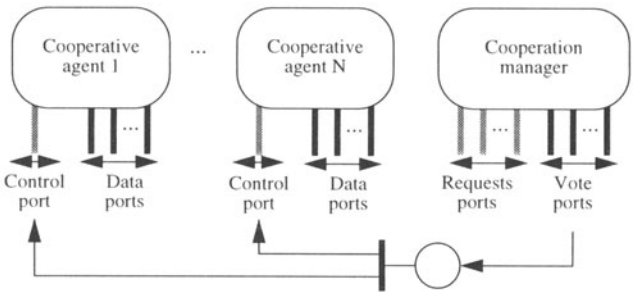

Figure 6 The VAL architecture of the specification

3.4 Protocol verification

The proposed protocol has been verified using the reachability graphs of different VAL specifications [VILL94].

The reachability graphs given by VAL have been analysed and no deadlock appeared. The graphs obtained are large, even for small cooperative groups. As an example, the reachability graph of a cooperation structure composed of three agents, whose valid instantaneous graphs retained are those containing at least two agents, gives a reachability graph of more than 10 thousands states and 31 thousands edges. As a consequence, their analysis has to be based on a reduction of the size using suitable projection technics.

A property to be verified is described using a VAL filter written in PROLOG syntax. The other transitions are ignored and transformed to have a neutral effect on the resulting graph. Once these operations on observable and unobservable events have been made, the obtained graph has to be reduced. This operation is done using the Aldebaran tool [FERN88], using observational and safety projections.

The following main properties have been checked:

Local properties (Projections on one agent), i.e. properties which concern the local behaviour of one cooperative agent in relation with the global evolution of the system. The way an agent evolves inside all the possible configurations of a cooperative group has been analysed, including:

- Cooperation membership (the behaviour of an agent when it requests to enter or to quit cooperation);
- Relations between agents (all the relations that an agent can have with other agents inside a cooperative group).

Global views of cooperation, i.e. projections that describe the global evolution of groups of agents inside the total cooperation. The projections presented aim to show the protocol behaviour for a cooperation structure and the information that the agents can have or can exchange inside a cooperation. They have been called global because they do no consider the behaviour of an agent for all cooperations, but they consider the behaviour of all cooperative agents for a particular cooperation structure, as:

- Valid instantaneous configurations (all different cooperative configurations that can be obtained);
- Cooperation phases (evolution of the cooperation to show the behaviour of the agents during the phases of the cooperation structure).

Let us note that it seems to be very time consuming to validate systems in which dynamic behaviour occurs.

3.5 Translation into an Estelle specification

The VAL specification has then been transformed into an Estelle specification (Figure 7) as it is easy to translate an Estelle [ISO9074], [BUDK87], [DIAZ89] specification into an implementation. Some VAL mechanisms have a direct equivalence in Estelle. Other mechanisms, as the multicast, that have no direct equivalence, must be transformed and adapted. Each of the VAL agents has been transformed into an Estelle protocol module instance. In fact, the VAL class of the cooperative agents becomes an Estelle module. One state of each VAL transition becomes an Estelle "major state". The other states of the VAL transitions are coded with internal PASCAL variables. The VAL transitions that contain multiple receptions have been divided into two Estelle transitions, the first one receiving the messages one by one within a loop for receptions, the second one processing all messages.

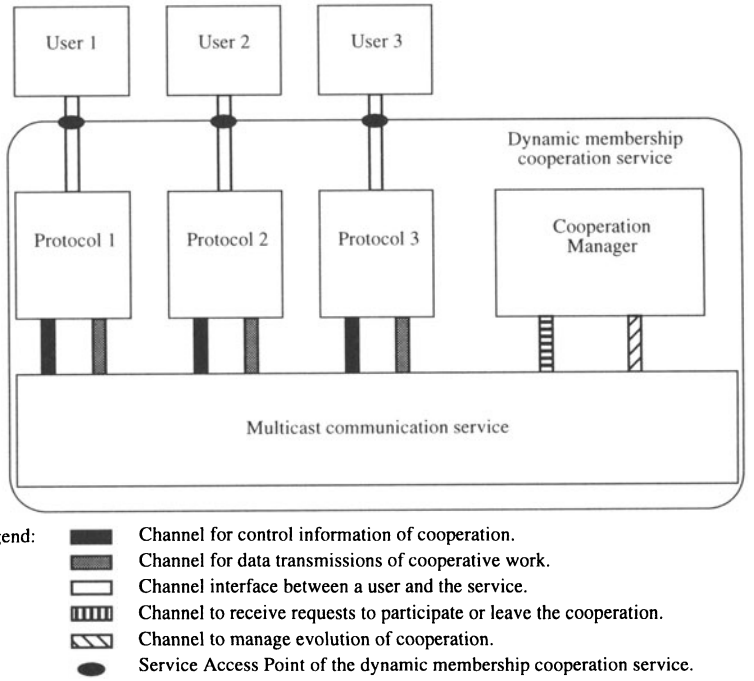

Figure 7 The Estelle specification architecture

The architecture of the Estelle specification (Figure 7) follows the layering principle of the OSI reference model [ZIMM80].

The Estelle code has been obtained using the environment EDT [EC92]. Moreover, it is possible to include inside an Estelle specification parts of C code and to call directly C functions. In an Estelle specification, modules are linked by bidirectional channels. In a real implementation, some of these channels are replaced by real communication supports that ensure data exchange between modules on different physical machines. The main limitation of this approach is related to the use of the temporary buffers for copying messages before external communication. Nevertheless, in the case considered here, the exchanged messages for controlling group membership are short, and the time lost is not significant. The implementation architecture has been built on a platform composed of SUN machines with the SOLARIS2. This implementation requires 6800 lines of Estelle and C code.

The chosen implementation can easily be adapted to other cooperations. Adapting it to new criteria means modifying the rules embedded inside the "Cooperation Manager" module. Defining a new cooperation means changing the structure of the graph, that is one array in each service module.

4 SERVICES AND PROTOCOLS FOR PRIVATE CONVERSATION SUBGROUPS

Let us consider an instant in which a set of agents are in cooperation, organized as a valid instantaneous graph. Inside this cooperation, it may be possible that a subset of the cooperating agents decides to conduct a private work, i.e. a private subgroup (PCS) that exchanges private data inside the PCS. As the same cooperation continues to run, the structure of this new PCS has been defined to be a subset of the structure of the current valid instantaneous graph.

The basic principles inside PCSs are the same as the ones proposed for global cooperations, as PCSs are some brief, temporary subgroups of the cooperating agents. Such subgroups may be of interest in many cases, as in teleconference, teleteaching, etc.

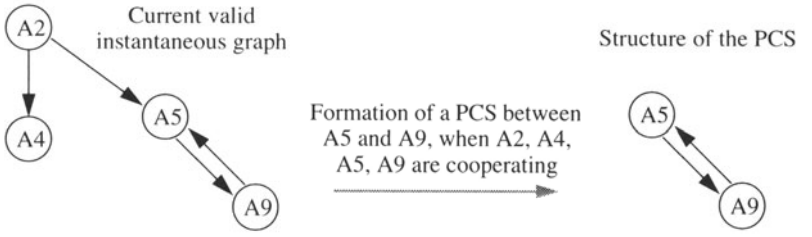

Figure 8 Structure of a PCS containing A5 and A9

Cooperating agents cannot refuse to belong to a PCS, but a not interested member can quit it immediately. An agent that belongs to a new created PCS has no initial data inside it and a PCS is closed when all its members have requested to quit it.

The hypothesis chosen can be considered to be too restrictive in comparison with more

general subgroups. They cannot apply and cannot being justified in the case of long-term subgroups created inside cooperative groups. Such long-term subgroups are considered as domains of cooperation [DIAZ94] embedded in a more general cooperation.

4.1 Service for PCSs

The service creates new PCSs inside cooperative groups, exchanges data inside PCSs and terminates the PCSs when they become useless. When a cooperating agent decides to form a new PCS, it gives to the service a list of agents which will belong to the new PCS. The service verifies if the PCS has not been created by another agent at this instant and refuses the creation if the PCS exists. During the evolution of the PCS, any set of members can decide to quit the PCS. For that, they indicate to the service their intention to quit and the service records their requests to exit. When all the members decide to quit the PCS, the service indicates the end of the PCS and the members effectively quit it.

When the problem of coherence of data is added to the life of a PCS, three main phases are obtained:

- the first phase corresponds to the creation of a new PCS. It consists of the request of creation and the reception of the indication of the creation of the new PCS.
- the second phase is the active phase of the PCS and corresponds to the exchange of data between the members.
- the third phase begins when all members want to quit the PCS. Then, the service suspends the exchanges of data inside the PCS and signals the end of the PCS.

4.2 Synchronisation with the group dynamicity

When the evolution of the structure of cooperation in time is considered together with the creation and deletion of PCSs, several problems arise. Suppose an agent creates a new PCS during a phase of deciding a change of cooperation. Moreover, suppose that the change is accepted. The agent has created a new PCS when the agents were linked by the old graph of cooperation. It could happen that some agents have quit the cooperation and that these agents were in the requested new PCS. Clearly, this is unacceptable; when an agent quits the cooperation, it must not be in a PCS.

Another problem arises with the PCSs which have been created before the change of cooperation. When an agent quits the cooperation and belongs to a PCS, the subgraph representing the PCS may become disconnected. Let us suppose that an agent that wants to quit the cooperation also belongs to a PCS. Let us suppose that the agent is not in the new graph of cooperation. Moreover, let us suppose that the change of cooperation is accepted. The agent quits the cooperation and so quits the PCS. But, when quitting the PCS, this agent can disconnect the subgraph representing the PCS.

These problems have been solved in forbidding the creation of new PCSs during any change of cooperation structure and in waiting for the end of old PCSs that can be disconnected by the change. Disconnectable PCSs can be arbitrarily closed by the service to avoid waiting too much time before the cooperation structure change.

4.3 Service and protocol specification

The proposed service has been described using the formal description technique Estelle [BUDK87], [DIAZ89] and some simulations conducted. A protocol has been developed and has been coupled to the previous dynamic cooperation service [DIAZ94]. The architecture of the whole specification is based on the layering principle of the OSI reference model. Each protocol entity is composed of a static Estelle submodule used for creating new PCSs. Another dynamic submodule is instantiated inside each agent that belongs to a new created PCS. The creations and destructions of PCSs are controlled by a PCS manager module directly coupled to the cooperation manager. This manager instantiates dynamically submodules that controls the new created PCS.

5 CONCLUSION

Cooperation has been seen as a conceptual level located on top of communication. This paper gives a starting basis for modelling and designing complex distributed systems. From intuitive definitions, it proposes a model for expressing some aspects of the semantics of cooperation, including: cooperation, four-valued predicates, domains of coherence, and inter-domain contradictions.

It has also presented the complete design of a service and a protocol that manage the dynamic structure of a cooperative group of agents in time. The specification and verification were first conducted using the VAL environment. Then, Estelle has been used up to the implementation. Verification methods based on the analysis of the reachability graph of the specification have to be improved as, even for small cooperative groups, the size of the reachability graphs increases very fast and future methods have to be developed for the verification of such services. For cooperative groups composed of agents that have similar behaviours, the reachability graphs obtained contain symmetries. This should is being used to handle symbolic reachability graphs to avoid developing the complete graph.

A service to form private conversation subgroups inside a cooperative group of agents has finally been added. But work has to go on to verify this service and to improve it in adding for instance error handling inside it before using it for a future implementation.

The management of the groups is centralized. To improve the robustness of the protocol, the distribution the group management responsibility is now studied. Current work consider the cooperation manager as a special token that circulates among the cooperative agents.

6 ACKNOWLEDGEMENTS

This study has been conducted inside the CNET-CNRS CESAME project on the design of high speed multimedia cooperative distributed systems, under grant 92 1B 178 from CNET FRANCE TELECOM.

7 BIBLIOGRAPHY

[AALS94] M. P. van de Aalst, K. M. van Hee and G. J. Houben. Modelling Workflow Management with High Level Petri Nets. Workshop on Computer Supported Cooperative Work, Petri Nets and related formalisms, Zaragoza, pages 31-50, June 1994.

[AGOS94] A. Agostini, G. de Michelis and K. Petruni. Keeping Workflow Models as Simple as Possible. Workshop on Computer Supported Cooperative Work, Petri Nets and related formalisms, Zaragoza, pages 11-29, June 1994.

[BANN89] L. J. Bannon et K. Schmidt. CSCW: Four Characters in Search of a Context. In Studies in Computer Supported Cooperative Work: Theory, Practice and Design, Eds. J. M. Bowers and S. D. Benford, Elsevier, 1991.

[BENF93] S. Benford and J. Palme. A Standard for OSI Group Communication. Computer Networks and ISDN systems, 25(1993):933-946. 1993.

[BLAI94] G. S. Blair and T. Rodden. The Opportunities and Challenges of CSCW. Journal of the Brazilian Computer Society, 1(1):3-14, July 1994.

[BUDK87] S. Budkowski and P. Dembinski. An Introduction to Estelle: A specification Language for Distributed Systems. Computer Networks and ISDN Systems, 14(1):3-23, 1987.

[DIAZ82] M. Diaz. Modeling and Analysis of Communication and Cooperation Protocols Using Petri Net Based Models. Computer Networks 6(6):419-442, December 1982.

[DIAZ89] M. Diaz and C. Vissers. SEDOS: Designing Open Distributed Systems. IEEE Software 6(6):24-33, Novembre 1989.

[DIAZ92] M. Diaz. A logical model of cooperation. Proceedings of the IEEE. Third Workshop on Future Trends of Distributed Computing Systems, pages 64-70. April 1992.

[DIAZ93a] M. Diaz. Coopération, Logique et Partage de Données. Proceedings of the AFCET and AFIA. Premières Journées Francophones Intelligence Artificielle Distribuée et Systèmes Multi-Agents, pages 253-262. April 1993.

[DIAZ93b] M. Diaz and T. Villemur. Membership Services and Protocols for Cooperative Frameworks of Processes. Computer Communications, 16(9):548-556. September 1993.

[DIAZ94] M. Diaz and T. Villemur. Formation of Private Conversation Subgroups in a Cooperative Group of Processes. Journal of the Brazilian Computer Society, 1(1):46-58, July 1994.

[EC92] EC ESTELLE to C compiler. Version 3.0. User reference manual. BULL S.A Copyright 1989, 1990. INT Copyright 1991, 1992., EDB ESTELLE simulator debugger. Version 3.0. User reference manual. BULL S.A Copyright 1989, 1990. INT Copyright 1991, 1992.

[EIJK88] P. Van Eijk, Ch. Vissers, M. Diaz Editors, The Formal Description Technique LOTOS, 1988, North- Holland

[ELLI91] C. Ellis, S. Gibbs and G. Rein. Groupware. Some issues and experiences. Communications of the ACM, 34(1):38-58. January 1991.

[EMER89] E. A. Emerson, J. Srivanasan, Branching Time Temporal Logic, Lecture Notes in Computer Science, Volume 354, 1989.

[FAGI85] R. Fagin, J. Y. Halpern. Belief, Awareness, and Limited Reasoning: Preliminary Report. 9th IJCAI 1985, Los Angeles, USA.

[FERN88] J.C. Fernandez. Aldébaran, Un système de vérification par réduction de processus communicants. PhD thesis, Université de Grenoble, France, 1988.

[HALP86] J. Y. Halpern. Reasoning about Knowledge: an Overview. Proceedings of the Conference of Theoretical Aspects of Reasoning about Knowledge, Ed. J. Y. Halpern, 1986.

[HEWI91] C. Hewitt and J. Inman. DAI Betwixt and Between: From "Intelligent Agents" to Open System Science. IEEE Transactions on Systems, Man, and Cybernetics, 21(6):1409-1419. November/December 1991.

[HUGH68] G. E. Hughes, M. J. Cresswell. An Introduction to Modal Logic. Methuen, 1968.

[ISO9074] ISO/IEC ISO 9074 : 1989 (E). Information processing systems. Open System Interconnection. Estelle: A formal description technique based on an extended state transition model, ISO IS 9074, Juin 1989.

[JUAN88] G. Juanole, O. Amyay, P. Azema, R. Gustavsson, B. Pershon. Towards a Knowledge-Base for Design of (N) Service-Protocol Pairs -An Epistemic Approach. In Research into Networks and Distributed Applications, Ed. R. Speth, North Holland, 1988.

[KARS94] A. Karsenty. Le collecticiel : de l'interaction homme-machine à la communication homme-machine-homme. Technique et science informatiques, 13(1):105-127, 1994.

[KRAE88] K. L. Kraemer and J. L. King. Computer-Based Systems for Cooperative Work and Group Decision Making. ACM Computing Surveys, 20(2):115-146. June 1988.

[KRAS91] H. Krasner, J. McInroy and D. B. Walz. Groupware Research and Technology Issues with Application to Software Process Management. IEEE Transactions on Systems, Man, and Cybernetics, 21(4):704-712, July/August 1991.

[MANN89] Z. Manna, A. Pnueli. The Anchored Version of the Temporal Framework, Lecture Notes in Computer Science, Volume 354, 1989.

[MILN80] R. Milner. CCS, a calculus for communicating systems, Lecture Notes in Computer Science 92, Springer Verlag 1980.

[SHOH89] Y. Shoham, Y. Moses. Belief as Defeasible Knowledge. 11th IJCAI 1989, Detroit, USA.

[VERN93] F. Vernadat and P. Azéma. Prototypage d'agents communicants. Proceedings of the AFCET and AFIA. Premières Journées Francophones Intelligence Artificielle Distribuée et Systèmes Multi-Agents, pages 129-140. April 1993.

[VICT89] F. Victor and E. Sommer. Supporting the design of Office Procedures in the DOMINO System. In Studies in Computer Supported Cooperative Work: Theory, Practice and Design, Eds. J. M. Bowers and S. D. Benford, Elsevier, 1991.

[VILL94] T. Villemur, M. Diaz, F. Vernadat and P. Azéma. Verification of Services and Protocols for Dynamic Membership to Cooperative Groups. Workshop on Computer Supported Cooperative Work, Petri Nets and related formalisms, Zaragoza, pages 73-91, June 1994.

[WERN88] E. Werner. Towards a Theory of Communication and Cooperation for Multiagent Planning. Proceedings of the 2nd Conference of Theoretical Aspects of Reasoning About Knowledge, Ed. M. Y. Vardi, pages 129-144, 1988.

[ZIMM80] H. ZIMMERMAN, OSI reference model, the ISO model of architecture for open systems interconnection, IEEE Trans. on Communications, vol. COM-28, April 1980.