

ABACO: COORDINATION MECHANISMS IN A MULTI-AGENT PERSPECTIVE

MONICA DIVITINI*, CARLA SIMONE[°], AND KJELD SCHMIDT[‡]

* Dept. of Computer Sciences, University of Milano, Via Comelico, 39/41,
I-20135 Milano, Italy. Phone: (+39) 2 55006313 Fax: (+39) 2 55006276 Email: divitini@hermes.mc.dsi.unimi.it

[°] Dept. of Computer Sciences, University of Torino, Corso Svizzera 185,
I-10148 Torino, Italy. Phone: (+39) 11 7429236 Fax: (+39) 11 751603 Email: simone@di.unito.it

[‡] Systems Analysis Dept., Risø National Laboratory, P.O. Box 49,
DK-4000 Roskilde, Denmark. Phone (+45) 46775146 Fax: (+45) 46755170 Email: kjeld.schmidt@risoe.dk

Abstract. The concept of Coordination Mechanism has been proposed as a means to construct computer-based facilities to support the articulation of cooperative work by employing a notation called Ariadne. The paper presents a mapping of the elements of Ariadne on the agent-based architecture called ABACO. This mapping serves two main purposes: it enables us to give a formal definition of the semantics of Ariadne and to define the overall structure of a crucial component of a general CSCW architecture that is specialized to support the construction of computational coordination mechanisms. The agent-based approach of ABACO allows us to satisfy the two basic requirements of Ariadne, namely malleability and linkability. Moreover, it allows us to conceive of how ABACO may incorporate functionalities so as to enhance the effectiveness of articulation work.

Résumé. La notion de Mécanisme de Coordination a été proposée comme un moyen pour construire des supports automatiques à l'articulation du travail, tout en employant une notation appelée Ariadne. L'article présente une correspondance entre les éléments composant Ariadne et les éléments d'une architecture d'agents appelée ABACO. Cette correspondance a deux buts: la définition formelle de la sémantique d'Ariadne et la définition de la structure d'un composant d'une architecture pour le CSCW, ce composant étant dédié à la construction des Mécanismes de Coordination. L'utilisation d'une architecture d'agents a permis de mettre en évidence deux propriétés fondamentales de ces mécanismes: l'adaptation et la possibilité d' être assemblés. Enfin, la notion d'agent permet l'introduction en ABACO de fonctionnalités destinées à améliorer l'efficacité de l'articulation du travail.

Field: Computer Supported Cooperative Work

Topics: Cooperation Models, Design Rationale Tools, CSCW Architectures

1. Introduction

The concept of coordination mechanisms has been proposed as a means for designing CSCW systems that support the articulation of cooperative work in the complex work environments that characterize modern industrial and administrative organizations (Schmidt and Simone, 1996).

A coordination mechanism is, simply put, a coordinative protocol with an accompanying artifact, such as, for instance, a standard operating procedure supported by a certain form. That is, we are talking about a phenomenon that is as ubiquitous as it is mundane.

As with any other procedure, the protocol of a coordination mechanism provides a 'precomputation' of activities which actors can rely on to reduce the space of possibilities (Norman, 1991). With coordination mechanisms, more specifically, it is the activities of articulating cooperative work for which a precomputation is provided. More than that, a

coordination mechanism is characterized by a specific and crucial relationship between protocol and artifact. On one hand, the protocol is objectified in the artifact. The artifact thereby gives a certain permanence to the protocol. That is, it conveys the stipulations of the protocol in a situation-independent manner. On the other hand, the artifact provides a representation of the state of the execution of the protocol and thereby serves as an *intermediary* between actors that mediates information about state changes to the protocol as it is being executed.

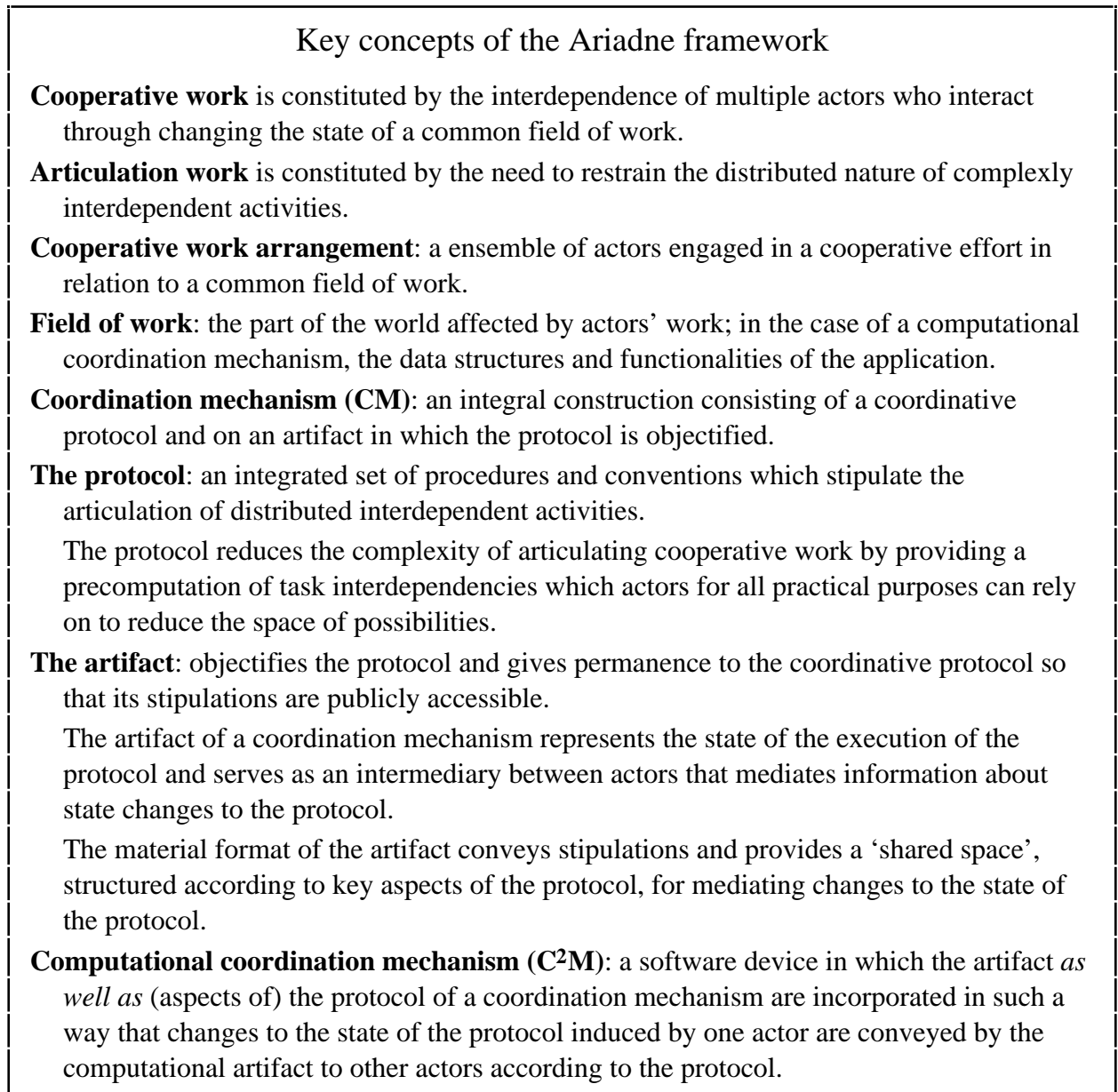


Figure 1. Key concepts of the Ariadne framework.

Consider the example of a bug report form. As an artifact, a bug report form, is a simple paper form with a number of labeled fields. A new bug report is initiated and filled-in by anyone involved in testing the software, which could be software designers, other designers, quality assurance staff, or marketing people. The originator of the bug report also provided a preliminary description and diagnosis of the problem. A so-called 'spec-team', that is, a group of three software designers responsible for diagnosing and classifying all reported bugs, then determines the module which presumably is responsible for the fault and, by

implication, identifies the designer responsible for that module and therefore for correcting the bug. The ‘spec-team’ also specifies project milestone by which time the bug should be corrected, and classifies the bug according to its perceived severity. Each designer is then responsible for fixing the problems (handling ‘his’ or ‘her’ bugs) and for reporting back to the designer who will be responsible for integrating and verifying software modules at the specified milestone. (Carstensen et al., 1995).

Let’s take a closer look at the interactions between the protocol and the artifact in this protocol-cum-artifact dyad. What we see is that *the state of the artifact changes according to the changing state of the protocol*. Firstly, the form is transferred from one actor to another and this *change of location* of the artifact in itself conveys, to the recipient, the stipulations of the protocol in a specified form, that is, the change of location transfers to the particular actor the specific responsibility of taking such actions on this particular bug that are appropriate according to the agreed-to protocol and other taken-for-granted conventions. Secondly, at each step in the execution of the protocol, *the form is annotated* and the thereby updated form retains and conveys this change to the state of the protocol to the other actors — the state of each reported bug is thus reflected in the successive inscriptions on the form made by different actors. That is, a change to the state of the protocol induced by one actor (a tester reporting a bug, for example) is conveyed to other actors by means of a visible and durable change to the artifact. In so far, the artifact can be said to provide a ‘shared space,’ a space with a particular structure that reflects salient features of the protocol.

Protocol-cum-artifact dyads such as the bug report form are specialized constructs and any given cooperative work arrangement will normally employ multiple coordination mechanisms. Typically the different protocols will intersect and must be aligned somehow and to some degree by the actors. When using the bug report form, for instance, participants will consult the project schedule to see the name of the designer that will be responsible for verifying a presumably corrected bug; this is implicitly indicated in the bug report form by the number of the platform period.

Though ubiquitous and mundane, conventional paper-based coordination mechanisms suffer from serious limitations with respect to supporting the articulation of the complexly interdependent and yet distributed activities of contemporary work settings. There are reasons to believe that computational coordination mechanism (C²M), i.e., a software device in which not only the artifact but also (aspects of) the protocol are incorporated in such a way that changes to the state of the protocol are conveyed by the artifact to other actors according to the protocol, can provide a degree of visibility and flexibility to articulation work that was unthinkable with previous technologies.

From a large corpus of empirical studies of uses of CMs for articulating cooperative activities we have derived a set of requirements for C²Ms and, by implication, for a notation for constructing them. The requirements can be summarized by two key-words:

- **malleability**: a C²M can be adapted to new demands in the course of its life-cycle (from its definition, through its instantiation, up to its activation), and
- **linkability**: a C²M can be linked to other C²Ms in a wider organizational context.

Ariadne is a general notation for constructing malleable and linkable C²Ms (Simone et al., 1995). Its main components are (1) a set of basic elements, called Objects of Articulation Work (OAW), which define a closed space of possibility from which to choose the elemental constituents of a C²M; (2) a set of basic relations to combine them into the protocols and

artifacts that constitute the C²Ms, and (3) a set of primitives guiding the users in defining, adapting, and linking C²Ms.

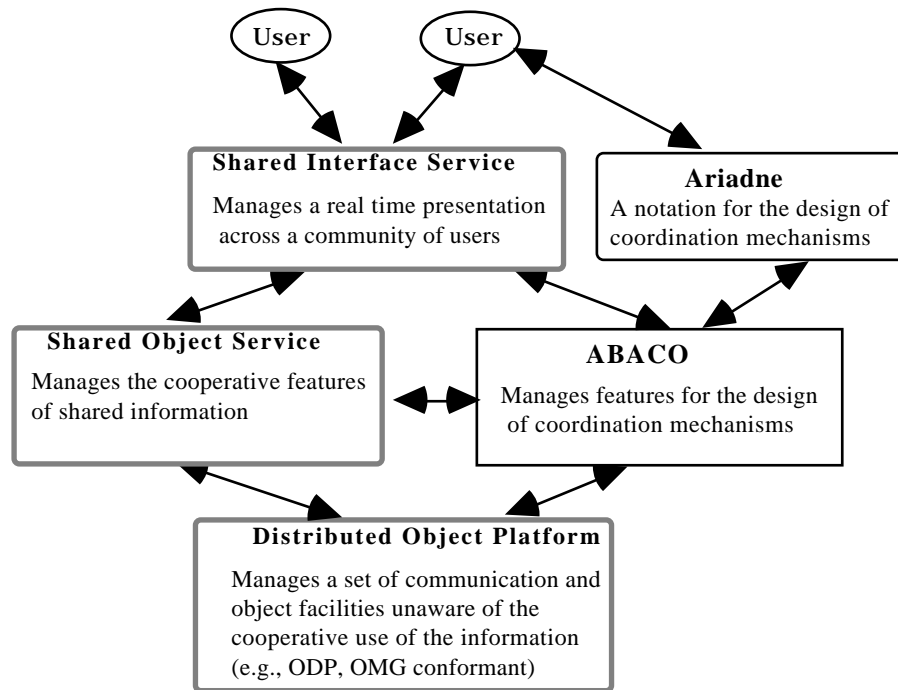


Figure 2. ABACO as a component of a proposed CSCW architecture.

The present paper describes the main characteristics of a framework, called ABACO (Agent-Based Architecture for COrdination mechanisms), where Ariadne can be implemented. One of the basic properties of ABACO is modularity, as a means to allow for malleability and linkability of the C²Ms to be designed through Ariadne. ABACO can be considered as a module of a CSCW platform which is specifically devoted to the construction of C²Ms, while employing the additional services of other modules. This idea is illustrated in Figure 2 in relation to the architecture proposed in (Trevor et al., 1994).

ABACO exhibits a multi-agent architecture organized into layers that mimic the conceptual structure of Ariadne. In very general terms, each component of Ariadne, from the basic elements up to the composite C²Ms obtained by the linking of more elemental C²Ms, is realized as an agent belonging to a specific type. Types of agents are characterized by their capability of communicating with the other agents that dynamically constitute their environment. Dynamicity is implied by the requirement of malleability and linkability of C²Ms.

The paper is organized as follows: The next section illustrates the communication requirements of Ariadne's components. Then the multi-layered agent-based architecture of ABACO is described together with its specific agent communication language, called Interoperability Language. In Section 4, this language is used to characterize the necessary agents types.

2. The communication requirements of Ariadne

As argued in the previous section, one of the main properties of Ariadne is compositionality, meaning by that the possibility of composing its elements in a flexible way to obtain the desired expressive power to define, adapt and link C²Ms.

Here, the components of Ariadne are presented in terms of the types of connection each component has to realize in order to be combined into broader components, and finally in the whole C²M. The types of connection define the communication requirements ABACO has to implement. Other aspects of the notation that are not pertinent to the definition of ABACO are neglected and can be found in (Simone et al., 1995).

Let us start with the *Objects of Articulation Work* (OAW), the most elementary components of Ariadne. The selection of this set of basic elements has been motivated elsewhere by presenting the so called Articulation Work Model (Schmidt and Simone, 1995). Ariadne contains the following OAWs: *Role*, *Actor*, *Task*, *Activity*, *Action*, *Interaction*, *Resource*. Here we focus on the list of their attributes with the related type that are derived from the above mentioned model.

The attributes characterizing each OAW can be classified as follows:

- a) attributes identifying the object and describing its informational content: they vary from an object to another and are not specified here, since they play no role in OAWs composition.
- b) attributes representing the reference to other OAWs or other C²Ms;
- c) attributes representing the relationships with the Organisational Context where OAWs are defined and adapted (c1) and where their behaviour is governed (c2), by possibly applying suitable policies. A policy is a set of rules or a reference to another C²M.
- d) The attribute *awareness* expresses the capability of the object to make other OAWs aware both of its internal state and of 'communication' received from its environment. It belongs to all objects and will no more be mentioned in their presentation.

In the description of the OAWs we adopt the following conventions: X^* denotes a finite sequence of items of sort X ; *Condition* denotes a boolean condition and *out-trigger* denotes a communicative event. Let us illustrate a selection of OAWs in terms of their different attributes.

Role =

ATTRIBUTE NAME	ATTRIBUTE TYPE
description	data-frame
responsible for	Resource*
responsible for	Task *
responsible for	C ² M*
involved in	C ² M*
precepts	set of rules
assumed by	Actor*
defined by	<Role, policy>
adapted by	<Role, policy>
awareness	<Condition, out-trigger>

Task =

ATTRIBUTE-NAME	ATTRIBUTE-TYPE
description	data-frame
preconditions	test
priority	value
postconditions	test
in-triggers	In-message*
under responsib. of	Role
supervised by	<Role, policy>
has allocated	Resource*
realised by	Activity*
criteria of accomplishment	<Role, policy>
assessed by	<Role, policy>
approved by	<Role, policy>
assigned by	<Role, policy>
defined by	<Role, policy>
adapted by	<Role, policy>
awareness	<Condition, out-trigger>

A *Role* is defined through a set of *responsibilities* (attributes of type b)) for *Tasks*, *Resources* and *C²Ms*. These responsibilities are established in the *Organisational Context* by the *Role* mentioned in the attribute *defined by* (type c1)). The definition of a *Role* can be changed by the *Role* mentioned in the attribute *adapted by* (type c1)), still in the *Organisational Context*. A *Role* can be *assumed by* one or more actors (attribute of type b)). The rules in the attribute *precepts* (attribute of type c1)) regulate the assumption of the *Role* by an *Actor* and the behaviour of the *Actor* that assumes the *Role*. For example, a *Role* can be assumed by people carrying a certain experience or possessing some formal property (like, a PhD) and its behaviour has to obey to some legal constraints.

A *Task* is characterised by its *description*, *preconditions*, *priority* and *postconditions*(all attributes of type a)). The *in-triggers* are communication events that must be listen to before the *Task* can start. Under the *responsibility of* specifies the *Role* that activates and accomplishes the *Task*, potentially making use of resources as described in the attribute *has allocated*. As a distributed entity, a *Task* is potentially realised by a set of *Activities* (all attributes of type b)). *Criteria of accomplishment* is a policy that the responsible *Role* follows to decide the termination of the *Task*, whereas the *Role* mentioned in the attribute *supervised by* must be contacted in case of exceptions. The *Roles* mentioned in the attributes *assessed by* and *approved by* are in charge of the declaration that the *Task* has been accomplished in a satisfactory way: these *Roles* can be expressed both in a absolute way or defined by a relation with the *Role* responsible for the *Task* (all attributes of type c2)). The *Role* in the attribute *assigned by* belongs to the *Organisational Context*: it assigns the responsibility of the *Task* to a *Role*. This attribute together with the remaining ones (see explanation of *Role*), are all attributes of type c1).

In the following, we just give a flavour of how the remaining OAWs are defined in the *Articulation Work Model*. An *Activity* is a structure of *Action* or *Interaction*, where a

structure denotes the causal relation binding the Actions constituting the Activity (this relations are out of the scope of the present paper).

Action and *Interaction* are characterized by attributes referring to other OAWs (type b)): for example in an *Action*, *initiated by* is the Actor starting the Activity the action is a part of, whereas the Actor mentioned in the attribute *done by* is the doer of the *Action*, which can be different from the previous one. Among the attributes characterizing an *Interaction* let us mention its *content*, *sender*, *receiver* and *illocutionary point* (Searle, 1975; Winograd and Flores, 1986). *Resources* are of different types: informational, material, technical and infrastructural. Their attributes vary in the different cases: however they all belong to the above mentioned types. The only aspects of a resource relevant here are the following. When a *Task* requires a *Resource*, its allocation must obey the *access rights* with the related policies. In a *Resource*, the attribute *governed by* refers to the Organisational Context, while the attribute *managed by* refers to the OAW level: the Role mentioned in this attribute is the Role responsible for the resource, whereas the Role mentioned in the attribute *governed by* is the one which assigns the responsibility of the resource to the responsible Role.

According to its definition a C²M contains an *Active Artifact* (AA) whose role is to mediate the articulation work among the cooperating entities involved in the C²M, either by notifying to them appropriate information in presence of particular conditions or by actively participating to the articulation work effort thanks to coordination capabilities.

Active Artifact =

ATTRIBUTE NAME	ATTRIBUTE TYPE
name	identifier
visibility	<Role, type>*
content	data-frame
update/read requests	<Role, request>*
coordination	<Condition, out-trigger> <In-trigger, function>*
defined by	<Role, policy>
adapted by	<Role, policy>
awareness	<Condition, out-trigger>

Proctor =

ATTRIBUTE NAME	ATTRIBUTE TYPE
name	identifier
defined by	<Role, policy>
adapted by	<Role, policy>
Description	Partial order relation OAWs × OAWs

The attribute *visibility* which assumes the value *update* or *read*, gives an account of the way each Role in the C²M can access the information contained in the attribute *content*. This latter can make reference to OAWs, e.g., in the case of C²Ms supporting classification schemes of the objects belonging to the field of work. The *update/read requests* are the communications the AA is prepared to receive: a request can be a *read* or an *update*, which has to be consistent with the type specified in the attribute *visibility*. *Coordination* and

awareness express AA's capability to propagate, among the C²M's components, information about changes to the state of the C²M induced by some of them. The condition triggering coordination and awareness can be based both on AA's internal states and on some communication event plus some constraints on the content of the received message: when the condition is satisfied, the AA sends a message which the receiver(s) must (coordination) or may (awareness) listen to. Accordingly, the message can be broadcast or sent to a destination explicitly specified in the out-triggers. Furthermore, the coordination can consist of the activation of some internal function as a consequence of the receiving of some messages which meet some conditions. Let us stress that coordination is a peculiar attribute of the AA: no OAW has communication of the type 'coordination'. The motivation relies on the fact that OAWs do not have, per se, an autonomous behaviour, rather they are just contributing to the definition of the overall C²M. Hence it is meaningless to coordinate them with other C²M behaviour. Instead, they are able to monitor their modifications and notify them as expressed by attribute awareness. In other words, OAWs possess a little degree of reactivity.

Conventions and procedures are represented in Ariadne through the notion of *Proctor*. A Proctor is a compound entity obtained from the composition of OAWs as in a cooperative arrangement conventions and procedures can be expressed in terms of relations among Tasks, Roles, Actors, Actions, Interactions and Resources. Proctors definition is the first example of application of the modularity characterizing Ariadne. For example, a Proctor can be obtained from the composition of Tasks and Interactions through the partial order relation appearing in its definition.

The last basic aspect of Ariadne refers to the linkability requirement, since as already anticipated, real working situations require a flexible combination of C²Ms to support articulation work. A comparative analysis of field study findings led to the identification of three basic modes by which C²Ms can be linked.

In *subscription mode* a C²M makes the behavior of another mechanisms part of its own behavior. For example, the subscription could involve the activation of another mechanism devoted to manage special types of negotiations or classification schemes as a way to access the resources of the field of work. An interesting case of subscription is when the definition of a Proctor can subscribe to other C²M in order to support a negotiation or in order to activate a process that is coordinated by another C²M. An example is when Proctors make reference to a C²M supporting conversations (Winograd and Flores, 1986) among Roles. This C²M is basically constituted by Interactions combined by causal relations and by an AA describing the status and history of the conversation. Through the reference to these types of C²M, the communication capability of Ariadne's components (and by consequence, of the Interoperability Language of ABACO, see next sections) is therefore very expressive and flexible in terms of patterns of interactions (Labrou and Finin, 1994).

In *inscription mode* a C²M provides information about its current state to another C²M (or, conversely, a C²M obtains information about the current state of another C²M). The inscription can be done in two ways: the reaction by the target C²M can be either *compulsory* or *voluntary*. In the case of compulsory inscription, the target C²M is expected to react accordingly and, if does not do so, then the compulsory inscription mode has to incorporate time-outs and solicitation in order to reduce the risk of (partial) blocking of the involved C²M. In the case of voluntary inscription, the reaction of the target C²M is voluntary in that it is provided with morsels of information that are supplementary to what is imperative. That is,

the voluntary inscription mode has to incorporate capabilities to allow the target C²M to voluntarily filter the provided information (Malone et al., 1987; Gasparotti and Simone, 1990; Fuchs et al., 1995).

In *prescription mode* a C²M over-writes the definition of the target C²M's behavior. This interaction mode allows Ariadne to honor the recursive nature of articulation work. In fact, in the prescription mode a given C²M can change the definition of another C²M, that is, the definition of its protocol, or its specification, for example, by enforcing a special state during its execution.

In order to support the definition of the links among C²Ms Ariadne employs the notion of Interface as a means to facilitate the construction of composed C²Ms: Interfaces manage the interaction among C²Ms. In particular the Interface associated to a C²M must specify the access rights to its AA and the possible interactions with other C²Ms with respect to the three linking modes. These modes are specialised into different attributes: in particular, the two types of inscription mode are called *awareness* and *coordination*, respectively. (The homonymy with the attributes of the OAWs and AA is not casual.)

C²M-Interface =

ATTRIBUTE NAME	ATTRIBUTE TYPE
name	identifier
External Visibility	<C ² M, AA-attribute, type>*
Subscription	<C ² M>*
Coordination	<C ² M in/out>*
Awareness	<C ² M, in/out>*
Control	<C ² M.Task*, pre-conditions, post-conditions>
Prescription	<C ² M, component*, in/out>

The attribute *External Visibility* states the type (read/update) of visibility of the current C²M's AA by the linked C²Ms: the visibility type can be different for each part of the data frame describing the AA.

The *Subscription* attribute describes the possibility for the C²M to interoperate with other C²Ms by activating their behaviour.

The *Prescription* attribute expresses the possibility of a C²M to be modified in one of its components by another C²M (in), or to modify (out) the definition of components of another C²M.

The *Coordination* and *Awareness* attributes describe the exchange of information with other C²Ms in the inscription mode; these attributes specify with which C²M the exchange is possible and in which direction (that is, if the current C²M is allowed to receive or send information). The *Control* attribute expresses a synchronisation among the execution of Proctors belonging to different C²Ms. For example, some Tasks of the current C²M must be synchronously executed with Tasks belonging to other C²Ms. This case can be viewed as a generalization of the coordination mode when the pattern of interaction is not fully specified but for the verification of some pre- or post-conditions governing the start and the termination of the synchronization.

3. The main features of ABACO

Ariadne allows the construction of a C²M by means of the elements of the notation described in the previous section. What has to be described now is the operational semantics associated to each of these elements. This goal is achieved by defining an agent based model of Ariadne, so as to obtain both the formal definition of its semantics and the overall structure of a software architecture, namely ABACO, where Ariadne can be implemented (Simone et al., 1995). We employ the notion of agent since it fits well with the main characteristic of Ariadne, namely compositionality through well established interfaces among elements. The actual implementation of ABACO is at the level of a demonstrator which served the purpose to check the main properties of the architecture.

The way in which agents are used in ABACO can be described in terms of weak agency as defined in (Wooldridge and Jennings, 1995). Following this notion "...the term agent is used to denote a hardware or software-based computer system that enjoys the following properties:

- autonomy: agents operate without the direct intervention of humans or others, and have some kind of control over their internal state;
- social ability: agents interact with other agents (and possibly humans) via some kind of agent-communication language;
- reactivity: agents perceive their environment and respond in a timely fashion to changes that occur in it;
- pro-activeness: agents do not simply act in response to their environment, they are able to exhibit goal-directed behaviour by taking the initiative... " .

This is probably the notion of agency more widely used outside the AI community. This latter employs a stronger notion according to which an agent is a computer system that has the previous properties and is moreover conceptualized using concepts usually applied to people, like mentalistic notions and rationality. In other words, AI defines agents as intentional systems (Dennet, 1987).

The two ways of conceiving agents identify two groups of agent languages, i.e. different computer systems which allow one to develop and program agent-based systems. As an example, in the first group we find ACTOR languages, TCL/TK and TELESRIPT; in the second one we find AGENT0/ PLACA and Concurrent METATEAM. A language that hardly fits in this classification is KQML/KIF. As stated in (Labrou and Finin, 1994), KQML is intended to be a universal interaction language that supports communication through explicit linguistic actions and that can be used in any environment where software agents need to communicate something more than pre-defined statements of facts. Although KQML is used in a number of projects, it still lacks a formal semantics. In (Cohen and Levesque, 1995) and (Labrou and Finin, 1994) there are some attempts to overcome this weakness.

From our perspective, the notion of weak agency has proved to be a very useful abstraction that allows us to conceive the implementation of Ariadne as a system constituted by different interacting agents organized at different levels of complexity. As for the communication language, the above mentioned proposals do not fit the requirements imposed by Ariadne that are strictly related to the semantic level of articulation work. Therefore the choice was to define a specific language and to postpone its possible mapping onto one of them when ABACO will be implemented in all its aspects.

3.1 The layered structure

Following the idea of realizing each element of Ariadne as an agent, ABACO exhibits a multi-layer architecture since C^2M s are compound entities that are built on top of the basic elements. In this section, we focus on the layers while the actual meaning of the arrows connecting the various agents (see Figure 3 and 4) will be explained in Section 4 once the Interoperability Language has been illustrated (Section 3.2).

Since OAWs are the building blocks of C^2M s made available by Ariadne the agents corresponding to them populate the first layer of the architecture. These agents are specialised to the management of the information characterising them, and of the relationships (communication) with their environment.

The second layer, related to each single C^2M , is populated by the agents capturing the communication capabilities of its Active Artifact (AA)¹ and Proctors. Figure 3 sketches the layout of a C^2M .

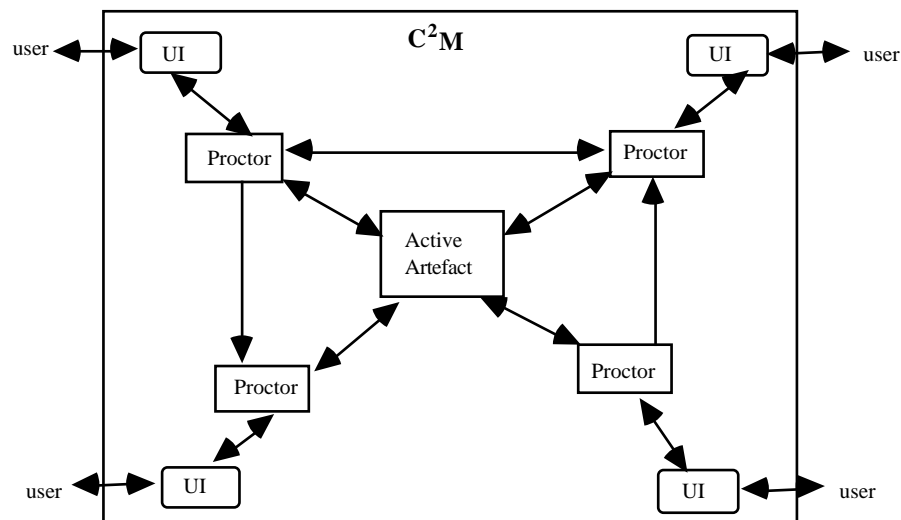


Figure 3. The agents constituting a C^2M agent.

A complete description of the User Interface agents (UI) it is out of the scope of our work. Here, it is sufficient to see them as agents responsible for communicating with the users (in the language of the chosen UI technology) and with the agents constituting C^2M s to convey to them the user's decisions and commands. In the opposite direction, the UI agent is the destination of the messages concerning user's awareness of what is going on inside C^2M s and through them in the field of work and the cooperative work arrangement. In this way, UI agents can naturally incorporate some standard services as filtering (Sheth and Maes, 1993) intelligent assistance (Greif, 1994) based on appropriate User Models (Kobsa and Wahlster, 1989) and on user's preferences. Notice that the ABACO allows the designer to define UIs specialized to each C^2M . This is a way to enhance the possibility to provide users with the more pertinent context in all activities they are involved in.

From the behavior point of view, a C^2M is the parallel composition of the behaviors of the related AA and Proctors on the basis of their communication capabilities.

¹ For the sake of simplicity, agents will be denoted by the name of the corresponding concepts.

Finally, the third layer is populated by the agents obtained from the composition of already existing C²M_s. The implementation of the composition of C²M_s is based on the definition of an Interface agent specialized to manage external communication (as depicted in Figure 4), and, in a more comprehensive view, specialized to support a first degree of tolerance of the modifications implied by the requirement of malleability.

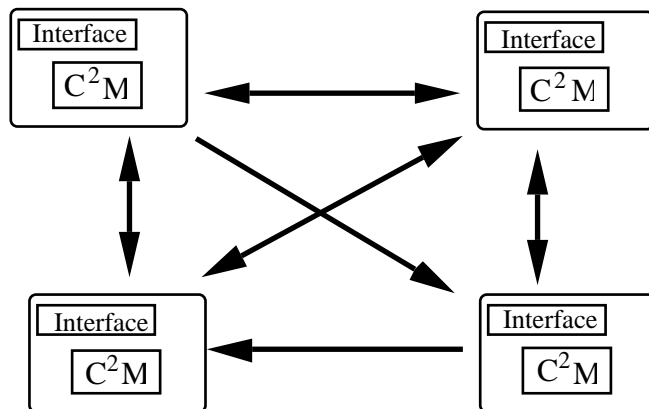


Figure 4 - Interactions between C²M_s.

The Interface agent plays the role of both facilitator and monitor of the embedded C²M in order to handle the additional communication needs derived from the composition of different C²M_s. Notice that the Interface agent can be employed in order to establish the type of interoperability between a C²M and the field of work whose activities the C²M is articulating (Genesereth and Ketchpel, 1994).

3.2 The Interoperability Language

In this section we describe the Interoperability Language (IL) that is used by the agents to interoperate with other agents that can also belong to others C²M_s. The three linking modes determine the basic primitives of the language. It is important to note that subscription and inscription modes are used by the C²M's components (objects of articulation work, artifact, proctors) to express their interactions. For example, an object of articulation work, for instance, a resource, can be accessed in the subscription mode to activate the policies governing its usage. The compulsory inscription mode (represented by the attribute coordination) typically expresses the reading from and writing to the artifact by the protocol in order to acquire and make visible imperative information. Finally, the voluntary inscription mode (represented by the attribute awareness) is typically used by the artifact to convey awareness of its internal changes to the Proctors. To the contrary, prescription mode relates (components of) a C²M, let's call it A, with (components of) another C²M, let's call it B. In fact, prescription expresses the cooperative handling of the modifications of A according to the protocol embodied in B. As a border-line case, B can be just a Role OAW. In fact, in this case the handling of the modifications can be seen as a degeneration of the previous case where the protocol and the related artifact collapse to a single person behaviour.

Therefore, the IL can be uniformly used in the architecture not only for formalizing the communication made explicit by the designer of C²M_s (for example, when an Interaction OAW is used to let two Proctors communicate) but also to 'implement' the implicit (that is,

system-defined) communication among all the elements of the notation (for example, when an OAW makes reference to another OAW in its attributes). In this way, the Interoperability Language is the basic means to realize the compositionality of Ariadne. That is, the interoperation of coordination mechanisms and of their components is described in a uniform way and, most importantly, is determined by the semantic level of articulation work.² This uniformity makes it possible to conceive of a flexible allocation of functionality between users and C²Ms, one of the basic aspects of malleability. Finally, from the technical point of view, the IL defines the operational semantics of Ariadne in a formal way, and allows us to complete the definition of the architecture of ABACO (see Section 4).

The IL contains primitives that take the following general forms

$$\begin{aligned} &\langle \text{Receiver} \rangle ! [\text{parameter}] \langle \text{message} \rangle \langle \text{ack} \rangle \\ &\langle \text{Sender} \rangle ? [\text{parameter}] \langle \text{message} \rangle \langle \text{ack} \rangle \end{aligned}$$

with the following meaning, respectively: the agent in whose specification the communication occurs sends to (!)/receives from (?) the $\langle \text{Receiver/Sender} \rangle$ the content $\langle \text{message} \rangle$; $\langle \text{ack} \rangle$ is a two-valued parameter: the default value ‘must’ specifies that the reaction by the target agent has to be compulsory, that is, it is required an acknowledgement from the receiver. The alternative value ‘may’ states that the reaction by the target agent is voluntary: it is not needed any acknowledgement from the receiver, who can use the received information or ignore it. $[\text{parameter}]$ can contain a value or be empty: it is used to specify which instances, among those associated to the Sender/Receiver, is involved in the communication.

Each $\langle \text{message} \rangle$ takes one of the following formats, where the operator $\langle X \rangle^*$ denotes a finite sequence of items of sort X:

- *tell*($\langle \text{variable:value} \rangle^*$): the sender wants to communicate to the receiver the values of some variables;
- *ask*($\langle \text{variable} \rangle^*$): the sender wants to get information into some variables
- *perform*($\langle \text{command} \rangle, \langle \text{parameter} \rangle^*$): the sender wants the receiver to perform the command with the given parameters;

$\langle \text{command} \rangle$ can be any of the following:

- $\langle \text{update} \rangle$ with parameters $\langle \text{variable:value} \rangle^*$, to assign values to variables
- $\langle \text{read} \rangle$ with parameters $\langle \text{variable} \rangle^*$, to read values into variables;
- $\langle \text{activate} \rangle$, without parameters, to activate the receiver agent; in particular the receiver can be a Task, an Action or a C²M. In the case of partial specification, the activated agent requires to the sender the missing parameters necessary to start its execution.

$\langle \text{participate} \rangle$ with parameter $\langle \text{Role} \rangle$; to involve the receiver in a C²M; the Role in the parameter is the one which activated the C²M.

- *synch*($\langle \text{CM.Proctor.Action/Task} \rangle^*$, $\langle \text{pre-conditions, post-conditions} \rangle$): each Action/Task belongs to a different C²M; this message implies that the Actions/Tasks have to be executed respecting the synchronization expressed by pre- and post-conditions.

- *over-write*($\langle \text{item} \rangle^*$, $\langle \text{new-item} \rangle^*$): the sender enforces modifications on some items in the receivers. When the first group of parameters takes the value ‘empty’, the command corresponds to the creation of the second group of items. Primitives containing an over-write

² This goes exactly the other way round with respect to the trend of applying programming language concepts to non-programming activities, like e.g., object-oriented approach.

message take into account the reflection (i.e., the recursive nature of articulation work) characterizing Ariadne (Schmidt and Simone, 1996).

Both in the case of a *perform*(read,...) and an *ask*(...), the sender wants to get the value of the specified variables. The main difference is that in the first case the sender knows about the structure of the data owned by the receiver and therefore the command must terminate successfully; in the second case, the variables denote an information structure owned by the sender where the command deposits the asked values, if and when it terminates successfully. An analogous distinction characterises the differences between *perform*(update,...) and *tell*(...).

The semantics of the IL primitives is defined in the same vein as the communication primitives in Hoare's CSP (Hoare, 1985), from which we also drew our inspiration to define their format. Basically, each primitive carrying a ! has to be executed synchronously with the corresponding primitive carrying a ?, and vice-versa, the correspondence being defined as the matching of the involved parameters. This choice is not restrictive since it is well known that asynchronous communication can be modeled by means of a pattern of synchronous communications with a process that plays the role of communication buffer. The modeling of this situation is out of our scope, and could be added to our framework by applying the techniques proposed, for example, by the Actor Model (Agha and Hewitt, 1987), under the same fairness hypothesis about the message handling system.

4. Agents communication in ABACO

The aim of this section is to employ the IL for the definition of the communicative behavior of the agents populating the three levels of the architecture illustrated in Section 3.1. The behaviors are constructed from the attributes characterizing the components of Ariadne (and by consequence, the communicative behaviour of the corresponding agents) by associating to them (scripts of) primitives of the IL. As discussed in the previous section, these attributes can be related (implicitly or explicitly) to the three linking modes. In this paper we consider just some attributes: a complete description can be found in (Divitini et al., 1995). We start from the attributes related to the communication within C²Ms, and then we illustrate the communication across C²Ms .

Communication among OAWs, AA and Proctors of the same C²M

1) *awareness* (mode: inscription voluntary)

The corresponding primitive of the IL is:

Int_condition AND ((X?message) AND (Msg_condition))*-->
 ||[X!tell(<variable:value>*), may]³

with the following meaning: if a condition on the internal state of the OAW/AA is satisfied (Int_condition) AND the OAW/AA receives one or more messages (X?message) AND each message satisfies the associated condition on its content (Msg_condition), then the

³ In this sequence messages can be combined using both the AND and OR connectors. In the first case the condition is satisfied only if all of the messages are received, in the latter the reception of one message is enough to trigger the events of the second part. The operator || denotes concurrent activation.

OAW sends (concurrently) one or more messages to other OAWs. Either the *Int_condition* or the set of incoming messages can be empty, but not both of them; the outgoing messages of the awareness conditions are sent with the *may* value of the *ack* parameter.

2) *coordination* (mode: inscription compulsory)

The corresponding primitive of the IL is:

```
Int_condition AND ((X?message) AND (Msg_condition))*-->
  |[X!tell(<variable:value>*)]
```

with the following meaning: if a condition on the internal state of the AA (that is on the values of its data-frame) is satisfied (*Int_condition*) AND the AA receives one or more messages (*X?message*) AND each message satisfies the associated condition on its content (*Msg_condition*), then the AA sends (concurrently) one or more messages to other agents (*X*). In this case either the set of incoming messages or the internal condition can be empty.

Within AAs coordination can take the following form too:

```
Int_condition AND ((X?message) AND (Msg_condition))* -->function
```

The left part can be described as above unless in this case the *Int_condition* can be omitted, whereas there must be at least an incoming message. The function triggered is a computation which involves the data-frame of the AA.

The communications related to coordination are characterized by the fact that the *ack* parameter has the *must* value both for incoming and outgoing messages.

3) *In-triggers* and the *Interaction OAW*

The *in-triggers* attribute (coordination mode) appearing in the Task represents an incoming communication that carries information that must be received before the Task can start. The corresponding primitive of the IL is:

```
X?tell(<variable:value>*).
```

The *Interaction OAW* expresses an explicit communication between Proctors and between Proctors and AA, either in coordination or in awareness mode. *Interaction OAWs* are represented as primitives of the IL containing a *tell* message whose parameters are directly derived from the attributes of the OAW (namely, sender, receiver, content, illocutionary point).

4) *activation, accomplishment, assessment and approval of Tasks* (mode: subscription)

A script involving several primitives of the IL realizes various attributes characterizing a Task, in the following way. The responsibility for a Task by a Role (under responsibility of) it associated to the following primitive:

```
Task!perform(activate)
```

If the Task needs some initial parameters to start the computation, it will ask them to the responsible Role: this will be its first operation.

The Role responsible for a Task will be waiting for a message:

```
Task?tell(return_parameters:value)
```

where *return_parameters* is a list containing the result of the computation; if it is empty, by this message the Task simply passes back the control to the Role.

Then the Role applies the `Criteria of accomplishment`, by possibly activating the related policy (see below).

When a Task has been accomplished, it must notify this event to the Roles which assess and approve it (`assessed by/approved by`), if any:

Role!tell(`return_value:value`).

Then the Task waits for an assessment and an approval:

Role?tell(`assessment:value`) and (Role?tell(`approval:value`))

and then either it terminates or it applies the policy possibly contained in 'value', when the approval fails.

5) *Attributes referring to OAWs* (mode: subscription)

As an example, we consider the allocation of Resources to Tasks.

When a Task needs a Resource, it must require its allocation through the following primitive:

Resource! perform (activate).

This command activates the check of the `Access rights` by applying the related policies.

Communication across C²Ms

As already mentioned, these communications are handled by the agent Interface associated to each C²M. Actually, when a communication involves an agent not belonging to the current C²M it is captured by the Interface that takes care of its handling. In this way, part of the communicative behaviour of the Interface is automatically defined by the attributes mentioned in the current C²M. In the following we consider some typical situations.

1) *awareness* and *coordination* (mode: inscription)

The description in terms of the IL is analogous as in the case of communication within C²Ms but for the interlocutor of the communication which takes the form:

C²M.Component

where Component is the appropriate agent in the external C²M. As usual, the parameter ack takes the two values may and must , respectively.

2) *defined by* / *adapted by* (mode : prescription)

The Organizational Context is, by definition, external to each C²M. Then the communications related to the prescription mode are filtered by the Interface of the C²M where the modification occurs.

The attributes `defined by` and `adapted by` are contained in OAWs, AA and Proctors and are associated to the following primitives, respectively:

<Role>?over-write(empty, Attribute*)

and <Role>?over-write(<Attribute, NewAttribute>*).

The effect of these messages is the definition and modification, respectively, of the attributes of the receiving OAW, AA and Proctor which are listed in the second parameter of

the primitive. Only modifications coming from the Role specified in the field `defined by/adapted by` are accepted by the OAW, AA or Proctor under concern.

3) *reference to policies /activation of and involvement in C²Ms*

(mode: subscription)

Another typical case of external communication is when the attributes involve a policy. Then the related C²M has to be activated by the Role mentioned together with the policy. To start a C²M the involved Role sends a message

C²M!perform(activate)

A Role can activate all the C²Ms it is `responsible for` plus a set of C²Ms that can be classified as publicly available, e.g. the C²Ms describing patterns of communications like conversations.

Then the activated C²M will send a request of participation to each Role it involves:

Role⁴!perform(participate, Role')

where Role' is the Role which started the C²M.

Furthermore, each Role is prepared to wait for a message from each C²M it is possibly involved in:

C²M?perform(participate, Role')

The role of the Interface in the management of the communication among agents belonging to different C²Ms described up to this point is based on the hypothesis that all the needed information is contained in the interoperating C²Ms, either implicitly or explicitly. This is a special case of interoperability which does not cover the more challenging situations in which the Interface is called to handle communication in the case of incomplete information. This is one of the open problems discussed in the concluding section.

5. Achievements and future Work

For the purpose of designing computational coordination facilities that support cooperating actors in managing the complexity of articulating their distributed activities, the multi-agent architecture ABACO has been proposed as a means to construct computational coordination mechanisms. The types of the constitutive agents and the characterization of their communication needs have been derived from Ariadne, a notation for the construction of computational coordination mechanisms, whose requirements and characteristics are based on empirical studies of uses of protocols for articulating cooperative activities. Specifically, the attention has been focused on the communication primitives devoted to the dynamic reconfiguration, cooperative control of propagation of changes, and interoperation within and across C²Ms.

At the present stage, ABACO is not defined in all details. However, the agent-based approach seems to be very promising for the following reasons.

⁴ Even if in the considered examples only roles can be involved in protocols, the architecture can be easily extended to consider a generic agent if needed.

The primitives of the IL can be used to express in a uniform way the interaction among the agents populating the three layers of ABACO. This is a strong property where to base the implementation of the malleability and linkability of the notation. This has been verified on the implemented demonstrator.

From the above point it follows that the functionalities currently available can be improved 'locally' by enhancing the capabilities of specific agent types, and compositionally employed in compound agents. The improvements we are envisaging show an increasing complexity.

First of all, the way in which the Interaction OAW is described (and implemented in ABACO) allows a direct integration of functionalities supporting the contexts in which communication occurs (Divitini and Simone, 1994b): specifically, the definition of User Models and Profiles in cooperative settings as proposed in (Divitini and Simone, 1994a).

Moreover, this proposal can be extended to cope with contexts pertinent to the procedures and conventions embodied in C²Ms through Ariadne (and consequently through ABACO). In other words, the proposed framework can be the basis for the definition of the notion of Group Models and Profiles since it provides a well established model of articulation work.

Finally, two aspects require a deeper investigation, both at the conceptual and technological levels. First of all, a more comprehensive definition of the notion of 'collaborative interoperability' (Mandiviwalla and Grillo, 1995), that in our terms is characterized by the semantic level of articulation work and by the presence of partial specification. Secondly, a deep understanding of how recursiveness of articulation work imposes structural requirements to the architecture where its support facilities have to be constructed. Some conceptualization of the notion of 'computational reflection' (Yonezawa, 1990) can be taken as a starting point to improve ABACO with the reflective features appropriate to the construction of computational coordination mechanisms supporting the recursiveness of articulation work.

Acknowledgements

The authors want to thank their partners in the ESPRIT BRA COMIC project (#6225). We acknowledge the contribution of Carla Quaranta Vogliotti and Monica Corigliano to the definition and implementation of ABACO.

References

- Agha, Gul and C. Hewitt (1987): Actors: A Conceptual Foundation for Concurrent Object-Oriented Programming. In *Research Directions in Object-Oriented Programming*, ed. B. Shriver and W. Wegner. Cambridge, MA: The MIT Press, pp. 49- 74.
- Carstensen, P. H., Sørensen, C., & Borstrøm, H. (1995). Two is Fine, Four is a Mess: Reducing Complexity of Articulation Work in Manufacturing. *COOP '95. International Workshop on the Design of Cooperative Systems, Antibes-Juan-les-Pins, France, 25-27 January 1995*. INRIA Sophia Antipolis, France, pp. 314-333.
- Cohen, Philip R. and Hector J. Levesque (1995): Communicative Actions for Artificial Agents. In *First International Conference on Multi-Agent Systems, San Francisco, CA, 12-14 June 1995*, ed. Victor Lesser. AAAI Press, pp. 65-73.
- Dennett, Daniel (1987): *The Intentional Stance*. Cambridge, MA: The MIT Press.

- Divitini, Monica and Carla Simone (1994a): Adaptivity in a system supporting cooperation. In *Fourth International Conference on User Modeling, Hyannis, MA, USA, 15- 19 August, 1994*, ed. B. Goodman, A. Kobsa, and D. Litman. User Modeling Inc., pp. 59-64.
- Divitini, Monica and Carla Simone (1994b): A Prototype for Providing Users with the Contexts of Cooperation. In *ECCE7, Bonn, Germany, September 5-8, 1994*, ed. R. Oppermann, S. Bagnara, and D. Benyon. GMD, pp. 253-270.
- Divitini, M., C. Simone, and C. Quaranta (1995): Coordination Mechanisms in a multi-agent perspective, ed. L. NavarroCOMIC, Esprit Basic Research Project 6225, Computing Department, Lancaster University. - [COMIC Deliverable 3.4. Available via anonymous FTP from ftp.comp.lancs.ac.uk].
- Fuchs, Ludwin, Uta Pankoke-Babatz, and Wolfgang Prinz (1995): Supporting Cooperative Awareness with Local Event Mechanisms: The GroupDesk System. In H. Marmolin, Y. Sundblad, and K. Schmidt (eds.): *ECSCW '95. Proceedings of the Third European Conference on Computer-Supported Cooperative Work, 10-14 September 1995, Stockholm, Sweden*. Dordrecht: Kluwer Academic Publishers, pp. 245-260.
- Gasparotti, P. and C. Simone (1990): A User Defined Environment for Handling Conversations. In S. Gibbs and A. Verrijin-Stuart (eds.): *Multi-User Interfaces and Applications* Elsevier Science Publishers.
- Genesereth, M.R. and S.P. Ketchpel (1994): Software Agents. *Communication of the ACM*, vol. 37, no. 7, pp. 48-54.
- Greif, Irene (1994): Desktop Agents in Group-Enabled Products. *Communications of the ACM*, vol. 37, no. 7, pp. 100-105.
- Hoare, C.A.R. (1985): *Communicating Sequential Processes*. Series in Computer Science. Englewood Cliff, NJ: Prentice Hall Inc.
- Kobsa, A. and W. Wahlster, ed. (1989): *User models in dialog systems*. Berlin, Heidelberg, Germany: Springer-Verlag.
- Labrou, Yannis and Tim Finin (1994): A semantics approach for KQML - a general purpose communication language for software agents. In *3rd International Conference on Information and Knowledge Management, CIKM'94, November, 1994*, .
- Malone, T. W., K. R. Grant, K. -Y. Lai, R. Rao, and D. Rosenblitt (1987): Semistructured messages are surprisingly useful for computer-supported coordination. *ACM Transactions on Office Information Systems*, vol. 5, no. 2, pp. 115-131.
- Mandiviwalla, M. and P. Grillo. 1995. TeamBox: An Exploration of Collaborative Interoperability. In *COOCS'95, Milpitas, CA, USA*, 347-353. ACM, Inc.
- Norman, D. A. (1991): Cognitive Artifacts. In Carroll, J. M. (ed.): *Designing Interaction. Psychology at the Human-Computer Interface*. Cambridge: Cambridge University Press, pp. 17-38.
- Schmidt, Kjeld and Carla Simone (1995): Mechanisms of Interaction: An Approach to CSCW Systems Design. In *COOP '95. International Workshop on the Design of Cooperative Systems, Antibes-Juan-les-Pins, France, 25-27 January 1995*, pp. 56-75.
- Schmidt, K. & Simone, C. (1996): Coordination mechanisms: Towards a conceptual foundation for CSCW systems design. *Computer Supported Cooperative Work (CSCW). An International Journal*, vol. 5, no. 2-3. - Forthcoming.
- Searle, John R. (1975): A Taxonomy of Illocutionary Acts. In *Language, Mind, and Knowledge*, ed. Keith Gunderson, vol. VII. Minneapolis: University of Minnesota Press, pp. 344-369.
- Sheth, B. and P. Maes (1993): Evolving agents for personalized information filtering. In *Ninth Conference On Artificial Intelligence for Applications*. IEEE Computer Society Press.
- Simone, C., Divitini, M., & Schmidt, K. (1995): A notation for malleable and interoperable coordination mechanisms for CSCW systems. In Comstock, N., Ellis, C., Kling, R., Mylopoulos, J., & Kaplan, S. (eds.): *COOCS '95. Conference on Organizational Computing Systems, Milpitas, California, August 13-16, 1995*. New York: ACM Press, pp. 44-54.
- Simone, Carla, Monica Divitini, Kjeld Schmidt, and Peter Carstensen (1995): A Multi-Agent Approach to the Design of Coordination Mechanisms. In V. Lesser (ed.): *Proceedings of the First International Conference on Multi-Agent Systems, San Francisco, Calif., USA, June 12-14, 1995*. Menlo Park, Calif.: AAI Press.

- Trevor, Jonathan, Tom Rodden, and John Mariani (1994): The Use of Adapters to Support Cooperative Sharing. In T. Malone (ed.): *CSCW '94. Proceedings of the Conference on Computer-Supported Cooperative Work, Chapel Hill, North Carolina, October 24-26, 1994*. New York, N.Y.: ACM Press, pp. 219-230.
- Winograd, Terry and Fernando Flores (1986): *Understanding Computers and Cognition: A New Foundation for Design*. Norwood, New Jersey: Ablex Publishing Corp.
- Wooldridge, Michael and Nicholas Jennings (1995): Intelligent Agents: Theory and Practice. *Knowledge Engineering Review*, vol. 10, no.2, 115- 140.
- Yonezawa, A. (ed) (1990): *ABCL: An Object-Oriented Concurrent System*. Cambridge, MA: MIT Press.