Journal of Universal Computer Science, vol. 3, no. 8 (1997), 865-898
submitted: 1/6/97, accepted: 10/8/97, appeared: 28/8/97 © Springer Pub. Co.

Ariadne: Supporting Coordination through a Flexible Use of the Knowledge on Work Processes

Carla Simone
(Dip. di Informatica, Universita' di Torino, Corso Svizzera 185, 10149 Torino, Italy
e-mail:simone@di.unito.it)

Monica Divitini
(IDI - Norwegian University of Science and Technology, Trondheim, Norway
e-mail: Monica.Divitini@idi.ntnu.no)

Abstract: Knowledge of the cooperative work processes characterizing an organization is a fundamental patrimony not only for people involved in their automation but also for the whole organization in performing its everyday activities. The paper focuses on *workflow technology* as a set of tools both supporting coordination and enhancing management of the knowledge of work and learning processes within a group of people coordinating their own activities. The paper presents a framework for the construction of coordination mechanisms whose design principles and tools make them a technology enabling the sharing of knowledge about processes and the incremental learning of people within the organization. These claims are illustrated through a working example.
Categories: H.1, H.4, I.2
Key Words: CSCW, coordination mechanisms, knowledge management, multi-agent systems

1 Introduction

Workflow technology has been proposed as a means to support cooperation in terms of coordination of activities and exchange of information and documents. Since its appearance it has been questioned because of its rigidity, inability to adapt flexibly to changes in the surrounding organization and relative demands. Thus from initial proposals focused on the definition of strictly formalized work processes (e.g., [Kreifelts et al. 1991a]) and the first generation of commercial workflow management systems) the trend shifted toward proposals based on the opposite idea: namely, questioning the suitability of work processes formalization, as if everything was an unique event needing to be re-defined every time it occurs (e.g., [Kreifelts et al. 1993], [Fuchs et al. 1995], [Trevor et al. 1993], [Fitzpatrick et al. 1996]). The latter proposals provide little support to process definition in the hypothesis that one shared working space can be sufficient. More recently, other authors have been tackling the problem by proposing specific process representations claiming that they are suitable for the users or for at least adequately supporting flexibility (e.g., [Shepherd et al.

1990], [Kaplan et al. 1992], [Malone et al. 1992], [Swenson et al. 1994], [Glance et al. 1996], [Dourish et al. 1996]).

We consider the above extreme positions equally inadequate in dealing with real user needs: this point has been raised and sharply discussed in [Bowers et al. 1995] and [Schmidt 1997]. Our position is somewhere in-between, not because we look for a simplistic or opportunistic compromise, but because we believe that serious consideration of the position supporting diverse user needs demands technology flexibility above all in relation to users' capabilities and interests. We examine this in greater depth here below.

First of all, the term "user" is often taken as a stereotype characterizing users either as naive actors incapable of doing anything with the work processes and related technology, or as skilled actors very efficient in organizing their work in cooperation with colleagues and in designing the related technology. Obviously, we are considering here not minor short-term activities but work processes where the coordination effort is constitutive by nature. Our first claim is that users are of different kinds, possess different skills and cultures, have different interests and attitudes, play different roles within the organization. Moreover, the above properties may be related to the work process at hand: that is, each (instance of) work process might define different values of these properties. Finally, the above properties can change according to the evolution of the user experience in participating in organization life. In other words, the users are embedded in a continuous learning process in which they participate with different moods, possibly in a discontinuous way, but in which they must participate if they want to feel part of the organization’s social system. Thus the technology, and in particular the systems supporting the coordination and execution of work processes, should be conceived of and designed neither as a prescriptive set of rules (possibly to be violated) nor as a tool to build ad hoc supports for temporary or specialized situations. On the contrary, it should be conceived of as a tool enabling the above mentioned learning process.

This is the specific viewpoint from which we approach the theme of knowledge management within organizations. In fact, we are aware that it is impossible to deal with all aspects at the same time: they are too many, involve too broad a range of disciplines and tools for managing their inherent complexity. We focus on the knowledge managed by people when dealing with the coordination of their work processes: a specific component of people's work that has been called *articulation work*. In so doing, we focus on how the technology supporting articulation work can serve as an enabler, among the other technologies and organization strategies, of the mentioned learning process.

The second point is that the technology can play this strategic role if it is based on some design principles that make it a flexible and useful tool.

By adopting the evocative terminology used in [Bowers et al. 1995], the introduction of workflow technology requires us to find a delicate balance between a *from within* and a *from without* the work process perspective in its design. The former perspective puts emphasis on the current work practices, the latter on the organization demands relative to what the authors call *(inter-organizational) accountability*. This consideration leads to the definition of a basic requirement for the technological support: namely, the capacity to take into account both perspectives and their interaction in a flexible way, that is, allowing for a dynamic shift of the borderline between the two. Our approach tries to fulfill this requirement by providing the means for *partial specification* and *incremental design* of the work process support.

These means have strong relationships with organizational knowledge management, in terms of both memory and learning. In regard to memory, partial specification allows the organization to record and make available the conditions of satisfaction a work process has to meet in order to be harmoniously inserted in its environment. These conditions are what make the actors safe in their work as they constitute a frame of reference where their autonomous choices are not source of problems for other organizational components. On the other hand, incremental design is what makes autonomous choices possible, because in addition to the specification of information that cannot be anticipated before the actual process instantiation it allows actors to select the operational behavior most suitable to the contingent situation according to the local working practices. We must emphasize that this view is not based on the unrealistic hypothesis that the interplay between constraints and autonomy is an easy achievement. To the contrary, as anticipated, it is the outcome of a dynamically adaptive process in which the necessary choices within the space of possibility induce a recurrent re-thinking, sometimes a re-negotiation, which is the stimulus and outcome of a continuous learning process.

An effective and timely shift of borderline between constraints and autonomy can be made possible by other two basic requirements: on the one hand, the tools for the partial definition and incremental design of the workflow support are under actors' control; on the other hand, this control is governed by the actual network of organizational responsibilities. In turn, these requirements imply the full *accessibility* and *visibility* of the tools by the actors, and the capability of the tools to take into account the dynamic relationships between the work process and its *organizational context*. The latter is another type of information entering the organization memory associated to the work processes. The main issue here concerns the way in which the organizational context plays an active role not only in the execution of the work processes but also in the evolution of their definition and instantiations.

Finally, the focus on the work practices and the need for continuous adaptation in the various interpretations illustrated above have an impact on the internal structure of the work process support. First of all, the workflow technology is based on the modularity and compositionality of the various work process components, at any level of granularity, in order to govern the modifications and their impacts. Secondly, these components are identified so as to make sense to the actors using them. That is, they are not generic (low level) objects; instead, they are *categories* at the semantic level *of articulation work*. Lastly, the technology supports the interoperability between components, at their different levels of aggregation, by providing *communication primitives* supporting both articulation work and interoperability.

These principles have been incorporated in a computational notation, called Ariadne; and in its implementation in a multi-agent architecture, called ABACO.

Through a working example the paper illustrates how the features of Ariadne can be used for designing coordination mechanisms able to enhance the management of knowledge related to processes and the process of learning within the group of people coordinating their activities. The paper ends with a short description of the implementation of Ariadne in ABACO.

2 ARIADNE: a Description of the Framework

The conceptual framework underlying Ariadne has been presented in [Schmidt and Simone 1996]. Ariadne's notation has been described in detail in [Simone et al. 1995] and [Divitini and Simone 1996], where a comparison with other approaches can be found. Since the focus here is on how Ariadne can be used in supporting users with varying needs and experiences within the organization, we briefly sum up the used terminology (see [Fig. 1]) and move on to how this framework can be used for the construction of flexible workflow systems.

Key concepts of the Ariadne framework

Cooperative work is constituted by the interdependence of multiple actors who interact through changing the state of a common field of work.

Articulation work is constituted by the need to restrain the distributed nature of complexly interdependent activities.

Cooperative work arrangement: an ensemble of actors engaged in a cooperative effort in relation to a common field of work.

Field of Work: the part of the world affected by actors' work; in the case of a computational coordination mechanism, the data structures and functionalities of the application.

Coordination Mechanism (CM): an integral construction consisting of a coordinative protocol and of an artifact in which the protocol is objectified.

The protocol: an integrated set of procedures and conventions which stipulate the articulation of distributed interdependent activities.

The protocol reduces the complexity of articulating cooperative work by providing a precomputation of task interdependencies which actors for all practical purposes can rely on to reduce the space of possibilities.

The artifact: is an information structure which objectifies the protocol and gives permanence to the coordinative protocol so that its stipulations are publicly accessible.

The artifact of a coordination mechanism represents the state of the execution of the protocol and serves as an intermediary between actors that mediates information about state changes to the protocol.

The material format of the artifact conveys stipulations and provides a 'shared space', structured according to key aspects of the protocol, for mediating changes to the state of the protocol.

Computational Coordination Mechanism (C^2M): a software device in which the artifact *as well as* (aspects of) the protocol of a coordination mechanism are incorporated in such a way that changes to the state of the protocol induced by one actor are conveyed by the computational artifact to other actors according to the protocol.

Figure 1: Key concepts of Ariadne framework [Divitini et al. 1996]

Ariadne provides the designer of workflow systems with features that are described in the following sections where the different elements of Ariadne will be presented in a frame-like notation. In the frames, words in italics denote concepts that will be described in subsequent sections.

2.1 The Notion of Coordination Mechanism

The notion of Coordination Mechanism (CM) is the unit of analysis of the target reality and the corresponding Computational Coordination Mechanism (C^2M) is the unit of technological support of the target workflow system. These notions represent a first step toward the fulfillment of the adaptability requirement which is at the basis of a creative use of the technology. First of all, the notion of Coordination Mechanism drives the designer to pay attention to the current work practices, both in terms of protocols and (paper based) artifacts actors are inventing to manage the complexity of the articulation of the activities they are involved in. Then the technological support is built on top of these practices. Secondly, the concept of C^2M allows the designer to tailor the target system into sub-systems in a way that makes sense to the actors as these sub-systems are identified from the work practices around artifacts. Thirdly, the inherent modularity and the related communication requirements give the target system a structure suitable for supporting its partial specification and incremental design.

According to the definition in [Fig. 1], a C^2M can be specified as follows:

Computational Coordination Mechanism =

ATTRIBUTE NAME	**ATTRIBUTE TYPE**
name	identifier
Active Artifact	*informational structure*
Protocol	*partial order relation* $\prod$ *CAW xCAW*
under the responsibility of	*Role*
defined by	<*Role*, policy>
adapted by	<*Role*, policy>

The attribute `under the responsibility of` specifies the Role responsible for the activation of the mechanism. The attributes `defined by, adapted by` specify who and in which way the mechanism is defined and modified. A policy can be either a set of rules or an invocation of another C^2M. These attributes define the organizational constraints to modifications. The type of the attributes `Active Artifact` and `Protocol` has to be understood as references to appropriate frames that will be presented later on.

2.2 The Categories of Articulation Work (CAWs)

The components of a C^2M are expressed in terms of Categories of Articulation Work (CAWs) (which are shown in [Fig. 2] together with their semantic relations); and of some Formal Relations which are expressed as a special interpretation of graphs to represent non-deterministic relations, and of partial-orders to express causal relations among CAWs. The set of Basic Elements was derived from studies of how artifactually imprinted protocols are designed and used by actors in everyday work activities, as reported in [Schmidt and Simone 1996]. They represent the minimal set of elements required to express the C2Ms examined in these field studies. However, it must not to be considered as definitive: it can be enriched if and when new needs emerge from the experimental use of the notation.

Distinguishing the CAWs according to their *nominal* and *actual* status (as indicated in [Fig. 2]) identifies categories pertaining respectively to the definition and specification of C2Ms. This distinction plays a role in adaptability: in fact, the *nominal* status defines the categories in order to express the constraints to what can be dynamically and incrementally specified by means of the *actual* status categories.

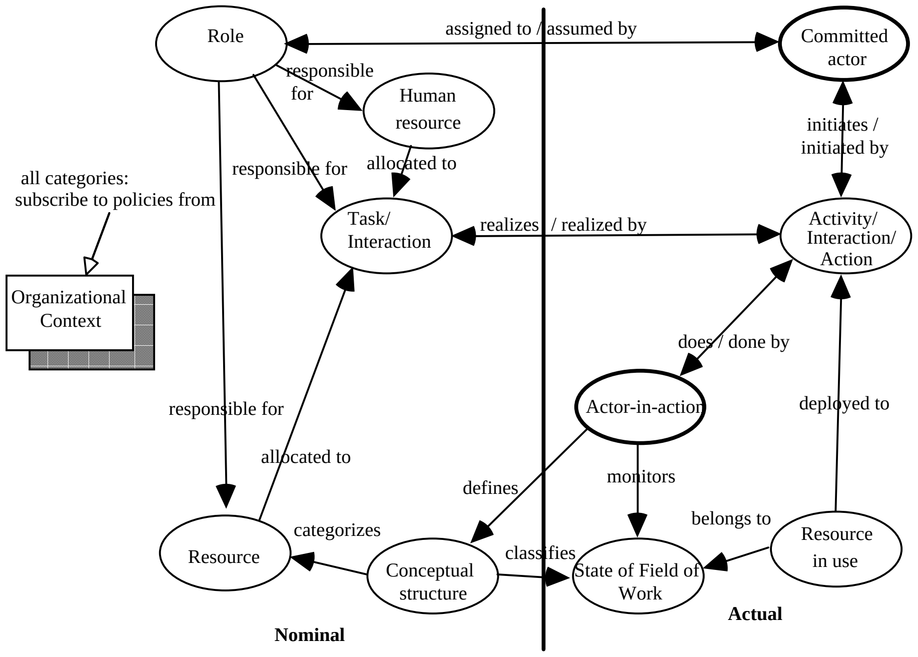


Figure 2: Ariadne's Categories of Articulation Work and their semantic relations

As for the Formal Relations, the field studies show that the possibility of representing relationships among entities is central to a notation for the design of CMs: e.g., causal relationships among tasks in workflows, part-of relations in classification schemes and so on. The literature provides several types of formalisms that serve exactly the purpose of expressing such relationships in an unambiguous way. At the current stage Ariadne contains the following relational structures: Labelled-Graphs, as a very general purpose formalism to represent non-deterministic relationships; Labelled-AND/OR Graphs and different classes of Labelled-Petri-Nets, as a way to represent causal relationships in presence of concurrency [Bernardinello and De Cindio 1992].

All are grounded in a sound mathematical theory providing algorithms for animation, simulation and analysis that can be exploited in the construction of a C2M.

All structures have associated labelling functions to express the interpretation of their constitutive elements, mainly in the set of Categories of Articulation Work (CAW). We briefly describe a class of Labelled-Petri-Nets that emphasizes modularity, namely Superposed Automata nets [De Cindio et al. 1982], since it will be used in the working example:

Labelled-SA-nets ::= [name_1: SM_1 ||.... || name_n : SM_n]

where name_i are names of objects that can be selected in the CAW (e.g., a Role); SM_i are Labelled-State Machines where only transitions carry a label; and finally, || is the parallel composition based on the synchronization of sending/receiving messages. Without entering into the details of SA-nets' semantics, we recall that the semantics of any concurrent systems can be different in relation to the strategy adopted in executing the sets of concurrent actions: *full concurrency* semantics when all possible actions are executed in one shot; *step* semantics when any subset of possible actions is executed in one shot. This subset can be identified by means of arbitrary criteria (priority, common property of the labels, and so on); and finally *interleaving semantics*: when just one action at the time is selected and executed in a fully non-deterministic way. Details about this and formal definitions can be found in [Pomello et al. 1992] in relation to Petri-net languages. These various semantics can be formulated equivalently in all formalisms representing concurrency.

2.3 An Environment for the Management of C^2Ms

The modularity (both in analysis and design) discussed in [Section 2.1] provides the designer with the possibility of considering processes with different characteristics in a systematic and coherent way. This means that each identified C^2M can be constructed by combining a protocol with different characteristics and eventually an artifact. While the modeling of the latter can be dealt with by exploiting classical data structures in combination with communication capabilities (see [Section 2.4]), the modeling of the protocol requires a more specific argumentation. In fact, all approaches to workflow modeling (either explicit or implicit into a specific technological solution) propose a single language, that is, a restricted set of basic categories and relations that allows the designer to represent workflows in a unique way: as a sequence of things to be done, as a flow of documents, as a negotiation of commitments, and so on. In some cases, the language allows us to consider resources too. Very often this makes the modeling activity a sort of translation from a hypothetical language closer to the target reality (for example, during the analysis phase) into the language imposed by the considered technology. Ariadne aims at providing the designer with the possibility of adopting within the same framework a variety of modeling approaches by combining CAWs and Formal Relations.
This richness has the obvious drawback of imposing an undue effort for whoever is defining each single C^2M, since the selection of the combination of components suitable for a specific work process is not an obvious task.

Hence the Ariadne interface is organized so as to reduce this overhead of effort by providing a place where such combinations (more formally called *grammars*) are defined, stored by and made available to the authorized actors.

When a user enters the Ariadne environment she has available three integrated levels (the central part of [Fig. 3]): one for the definition and modification of grammars (γ level), one for the construction of C2Ms in terms of protocols and artifacts (β level), and one for the activation of the C2Ms (α level). Each framework is tailored to the specific needs of its potential users.

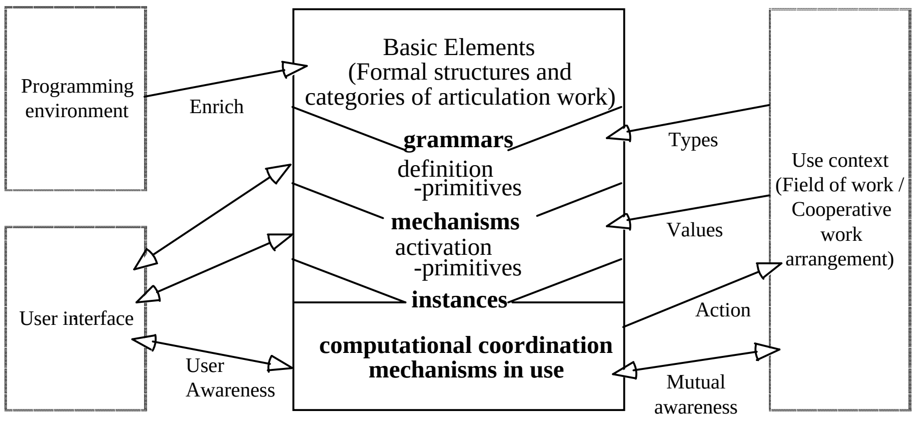

Programming environment
Enrich
Basic Elements (Formal structures and categories of articulation work)
grammars
definition -primitives
mechanisms
activation -primitives
instances
computational coordination mechanisms in use
User interface
User Awareness
Types
Values
Action
Mutual awareness
Use context (Field of work / Cooperative work arrangement)

Figure 3: The structure of the Ariadne environment and its contexts of development and use

Building a grammar, at the γ level, means determining the expressive power of a language for defining a class of C2Ms as well as the operational semantics associated to the elements of the grammar. The 'space of possibility' within which grammars can be defined at this level is determined by the set of categories of articulation work and by the available Formal Relations (the Basic Elements of [Fig. 3]).

As an example, let us consider the definition of a grammar [1], called CONV_GR, for the construction of different types of conversation models [Winograd and Flores 1986] that are traditionally described by means of a Labelled Graph. Then the definition of CONV_GR assigns to the protocol an L-Graph whose arcs are labelled in the set of Interactions.

```
CONV-GR := (// protocol)*
protocol := Role <-- structure of Interaction
```

1 The definition of grammars is given in terms of production rules. The starting symbol is always the name of the grammar while the other non-terminals are taken in the set of component of the notation. // denotes parallelism; X* denotes an arbitrary sequence of elements of type X.

```
structure := L-Graph
```

This is realized by making available an interface where L-Graphs can be edited and the related arcs labelled in the set of the Interactions. The arc labelling function is not arbitrary. Indeed, it has to follow some semantic constraints: for example, the fact that an *accept*, *counteroffer*... must follow a *request* /*offer*. This property can be represented in the notation by suitable predicates that are defined together with the grammar.

As a second example, let us consider the definition of a grammar called WF_GR for the construction of workflows that the designer wants to represent by a formalism describing distributed states and actions. Then, the designer chooses to base the description on Superposed Automata nets.

WF_GR accesses a framework where Labelled-SA-nets can be edited and their transitions are labelled either as a Task or an Interaction. The names of the constituting State Machines are defined as the name of a Role extended by the acronym of the current CM. The formal specification of the grammar is as follows:

```
GR-WF := (// protocol)* // Active Artifact
protocol := Role <-- structure of Task/Interaction
structure := Labelled-SA-nets
Active Artifact/Role/Task/Interaction := see related frames.
```

At the β level, the user can then define (or modify) the C2M itself according to the chosen grammar. In this context, the user can determine the allocation of functionalities between human actors and C2M, select the degree of partial specifications, and make permanent changes to an existing C2M as part of its evolutionary design.

Finally, at the α level, the user can instantiate and activate the C2M in a particular situation and do so in an incremental fashion; moreover, at this level the user can make local changes to the C2M instances in order to deal with ad hoc needs.

While the two levels of definition and activation of C2Ms can be recognized in almost all recent CSCW applications, the first level, where it is possible to define grammars, is unique and allows for the desired flexibility in work process modeling. The different levels of Ariadne are typically accessed by users with different skills and necessity. At the β and α levels, the use of the notation merely requires the ability to select and combine predefined items according to the rules of a relevant grammar and the associated semantics. These levels are typically needed by end-users who, as part of their everyday work activities, use, adapt and in some circumstances define coordination mechanisms. The γ level, on the other hand, is typically the realm of the 'application designer' or, in our framework, of actors who define grammars needed by a particular community of end-users for defining their protocols.

The full visibility and accessibility of the three levels, while preserving their specific context and functionality, is the basis of what has been proposed as an incremental approach to customization [MacLean et al. 1990]. Thus, in the following, user and designer are to some extent synonyms.

2.4 The Role of Communication Features

In Ariadne the role of communication is crucial in many ways. First of all, the categories of articulation work and the active artifact are equipped with communication features allowing them to show a reactive and proactive behavior. Let us consider how an active artifact is specified:

Active Artifact:

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

The attribute `visibility` specifies the access rights of the roles involved in the C^{2}M. The attributes `update/read requests`, `coordination` and `awareness` specify the actual behavior of the artifact and in which sense it is *active*. In particular, `coordination` and `awareness` deserve some explanation. The first attribute tells under which condition or incoming communication the artifact reacts either by issuing a communication or starting an internal data manipulation. These activities are mandatory, in the sense that they are a constitutive part of the protocol the artifact is participating in as a 'partner'. On the other hand, the communication associated to the attribute `awareness` is used to allow the artifact to participate in the construction of the so called 'awareness information' which is not intrinsically part of the protocol but is recognized as playing a relevant role in effective coordination. Awareness information can be filtered in input and in output as specified by the guards expressed by the conditions. The attributes `coordination` and `awareness` appear with the same purpose in all CAWs' specification.

Secondly, the distributed nature of the protocols contained in the C^{2}Ms requires communication capabilities that can be expressed by the CAW called Interaction. The next section will illustrate this aspect in depth.

Finally, the mentioned possibility of making C^{2}Ms interoperate is based on the communication across C^{2}Ms that characterizes their Interfaces. Actually, the modularity of Ariadne and its features allowing for communication among and across its components at any level of aggregation is implemented through a multi-layered architecture of agents whose behavior is characterized by their communication capabilities [Divitini et al. 1996]. In this architecture, which will be described in

[Section 5], C^2Ms interfaces play the fundamental role of filtering and managing the communication across C^2Ms.

3 Ariadne at Work: a Working Example

The working example is a simplified version of a real case presented in greater detail in [Divitini and Simone 1996]: here the details that are not essential or relevant in this context are omitted, since the description is functional to an intuitive presentation of the different elements of the notation and of its use in supporting cooperation. We start with the presentation of the scenario and then we simulate the construction and activation of the required C^2M.

3.1 The Scenario

In the following we will analyze two coordination mechanisms that we have observed in a organization, let's call it Alfa, whose mission is to promote the constitution of consortia for the development of targeted projects.
The development of the projects is achieved through a set of processes of different nature that are both autonomous and coordinated. When Alfa was established, the only computer-based support provided to its members was a simple communication system (basically, e-mail and moderated conferences) connecting the various people. This solution was acceptable at the beginning but became inadequate with the increase in size of the organization, in terms of numbers of both members and sponsored projects. Then the problem was to find a technological framework where the various ongoing processes, and in particular their interactions, could be supported. In fact, the loose organizational structure characterizing Alfa was the source of a low cohesion of its members and of a reduced mutual visibility. We will consider in this paper how Ariadne has been applied for the construction of a mechanism supporting the management of the various projects and the related research area.
Each project that is approved by the devoted committee is classified as belonging to a specific research area. Each research area is under the responsibility of a Research Area Coordinator (RAC). The management of a research area and of the related projects has many obvious goals, among which we will consider only what is related to the timely submission of the project reports and the implementation of the decisions taken during the periodical review meetings of the above mentioned committee. When Alfa was established, the management of an area was seen only as the juxtaposition of the management of the single projects belonging to it. At this stage the RAC was responsible simply for delivering the decisions of the committee to the appropriate Project Leader (PL) and for handling exceptional situations that could arise during the development of one of the projects belonging to her area. As the complexity of the projects increased, the RAC became more operative, assuming relevant tasks concerning the coordination of the management of the projects: among

the others, the responsibility for the preparation of the project reports that have to be delivered to the committee in order to verify the status of a project. In fact, one of the recognized difficulties in the review process concerned the different levels of quality and timely delivery of the reports: this made their comparative evaluation difficult.

Both Project Management and the Research Area Management work processes are obviously quite complex and could be analyzed from different points of view and supported in many different aspects. As already said, the focus here is mainly on the interaction protocols needed to synchronize the action with the activities performed in the Review Meeting process: the production of the documents to be approved and the management of the impacts of the decisions taken during the review. As project teams are composed by several actors, loosely coupled in terms of organizational structure, the problem was to define a protocol among them which could guarantee the timely propagation of the decisions and the consistency of the distributed actions according to them. Then, a 'detailed map' should be drawn, to serve as a reference point for all the involved roles.

In the following subsections we will describe the project management in greater detail in order to exemplify the various elements of the notation. Description of the research area management will instead appear in [Section 4]. There we will see how Ariadne can support its users in introducing new mechanisms for overcoming the inadequacies of existing ones in dealing with a changing situation.

3.2 The Construction of a C^{2}M

In order to construct a C^{2}M the user has to specify the requisite CAWs and select the appropriate grammar.

CAWs are specified by instantiating the templates provided by the notation. Let's consider, as an example, the roles involved in the described scenario. The project management involves a project leader and a variable number of designers who can be classified as senior or junior. Senior designers assist the project leader and are in charge of the revision of specification, while junior designers have a more "operative" role.

In Ariadne we can specify these three roles starting from the template that it provides, described below.

Role =

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

Each `role` is defined through a set of `responsibilities` for `tasks`, `resources` and C^2Ms. In the considered scenario, the role Senior Designer, for example, is responsible for revising the specification and appointing their implementation to a Junior Designer, for assisting the Project Leader in planning and for working with Junior Designers in the preparation of the required reports. The Project Leader is responsible for the whole mechanism. Role responsibilities are established in the Organizational Context by the `role` mentioned in the attribute `defined by`. The definition of a `role` can be changed by the `role` mentioned in the attribute `adapted by`, still in the Organizational Context. A `role` can be `assumed by` one or more `actors`. The rules in the attribute `precepts` regulate the assumption of the `role` by an `actor` and the behavior of the `actor` that assumes the `role`. For example, a `role` can be assumed only by people carrying a certain experience or possessing some formal property (like, a PhD) and its behavior has to obey some legal constraints. In the case under analysis, for example, the role of Project Leader is normally assumed by the main contractor of the consortium, and by default by the Research Area Coordinator.

The definition of a role, as well as the one of the other categories, can proceed in an incremental fashion and can be interleaved with the selection of the grammar and the definition of the needed protocols and active artifact.

The grammar is selected by using the primitive *access*(Grammar-set), where Grammar-set contains the grammars constructed at the γ-level. If we consider the grammars defined in [Section 2.3], then Grammar-set = {WF_GR, CONV_GR}. If no adequate grammar is available, the user can (ask some authorized person to) define a new grammar using the primitive *define-Grammar*(X), where X is the name of the new grammar.

The analysis of the work processes in Alfa led to the selection of the grammar WF-GR. In fact, it was evident that the overall structure of the Project Management is influenced by the presence of a paper based artifact the project teams introduced to support coordination among participants in the project. When computerizing the coordination mechanism, it was decided to let the computer based artifact manage the same type of information and in addition assume an active role in promoting coordination and awareness. Then, the `data frame` of the `active artifact` is a structure of project modules: information about each module is expressed by some attributes with the related access rights. For example, the attribute reporting the state of the project can be updated only by the Project Leader; Junior Designers have access to all the pieces of information recorded in the artifact but cannot change them. Its communication capabilities make the `active artifact` an active component contributing to the protocol supporting the articulation work necessary to manage (the considered aspects) of the project management.

The protocol coordinating the members of the project team was naturally expressed by a structured and distributed flow of `tasks` and `interactions` across the involved `roles`, in order to represent not only the related responsibilities but also the causal order in which they have to be fulfilled. Then, a formal structure is needed for representing this relation. It is important to notice that Interaction is a Category of Articulation Work that does not simply model the exchange of a message among the various components of the mechanism, but allows definition of specific communicative protocols, possibly activating devoted mechanisms. For example, if necessary it is possible to activate a conversational protocol [Winograd and Flores 1986] (constructed by means of CONV-GR) for dealing with the undertaking of a commitment between the communicating actors.

The designer can define all the above mentioned components of the C^2M by means of the primitive *define-C^2M*(X), where X is the name of a C^2M. Specifically, the protocol, including the communication capabilities of the active artifact, is visualized [2] in [Fig. 4] with the following conventions: for sake of conciseness, all message content implicitly refers to the project the PL is leader of; all labels are either `tasks` that can be described in detail as in the case of the other CAWs, or `interactions`; synch (Roles, info) denotes a special kind of `task` which has to be performed jointly by the mentioned `roles` and has to produce the mentioned `resource`. The way in which this cooperation is performed can be established, dynamically and/or incrementally, by specifying the `task` attributes such as the activities realizing the cooperation or by the activation of a specific protocol, i.e., of another C^2M. By default, synch (...) opens an interaction space where the joint behavior can be performed. Finally, CALL(...) denotes the invocation of a function of the Field of Work that is not specified here.

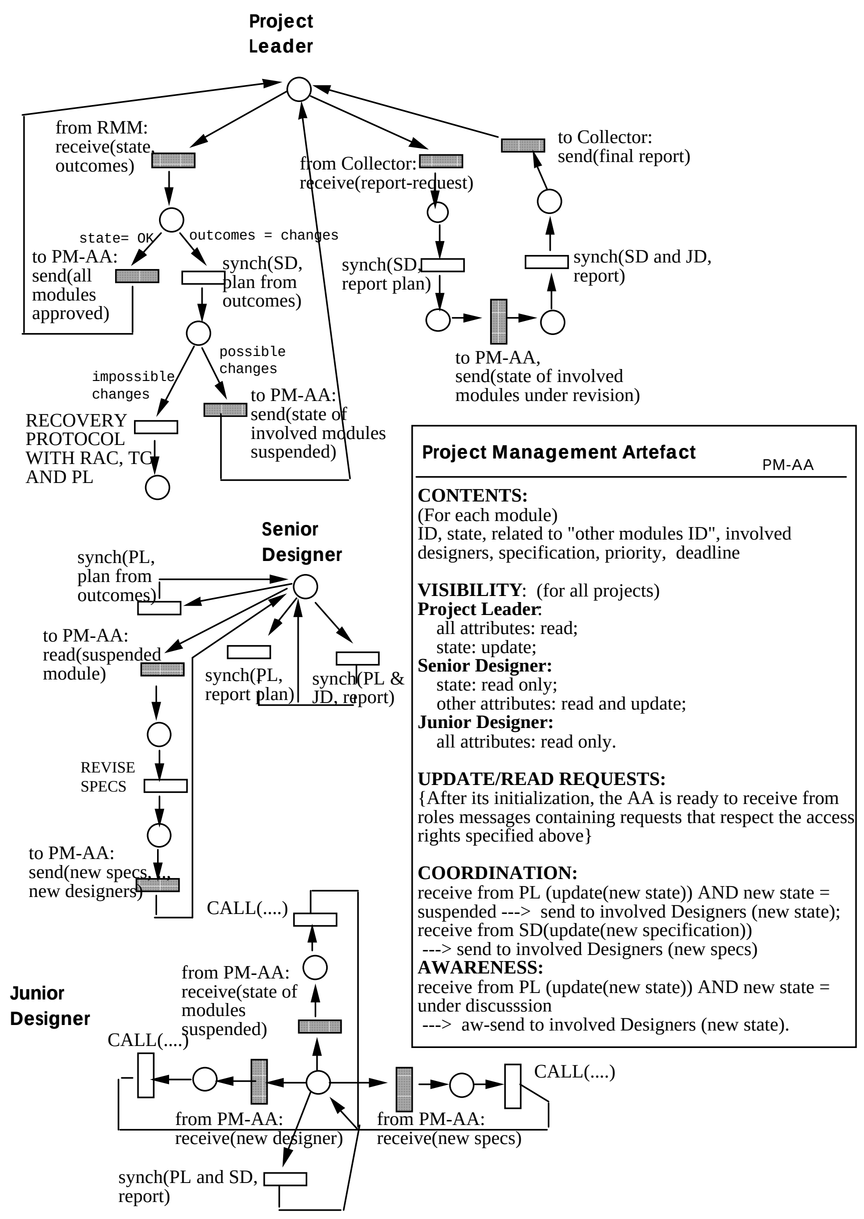
Once the C^2M is defined, two primitives help the designer in verifying its properties: *animate*(X) and *simulate*(X), where X is the name of the C^2M. These primitives provide the user with the possibility of ‘playing’ with the C^2M for verifying its behavior. Simulation is an animation with the additional computation of predefined parameters expressing good performances. These functions have an

2 The use of this formal language is just limited to giving a graphical representation of the causal relations to complete the conventional way of describing C^2Ms. This does not mean that this language is proposed to visualize them in the real user interface.

obvious value during the definition of the C^2M because they allow the designer to test different possibilities and then choose the best one. Moreover, they can be successfully exploited also for increasing the awareness of the actors potentially involved in the execution of the mechanisms. In fact, through an animation (or a simulation) the users of the mechanism can familiarize with it before actually starting to act within the organization.

This is a very good way for newcomers to learn the protocols that are established in the group they are entering and to understand how their actions are going to impact on the overall cooperative effort.

Another important primitive is provided by Ariadne at this level: *access*(C^2M-set). This allows users to access to the mechanisms that have already been defined, of course respecting the privacy criteria that are defined through adequate policies. The availability of this kind of knowledge is essential for users when they have to define a new mechanism. In fact, the C^2M-set together with the history of their evolution managed by the primitive *history*(X) where X is a C^2M name, constitutes an elementary form of organizational memory on processes and, as such, it allows members of an organization to take advantage of the experience gained by other people. Even if there is no mechanism that can be exploited directly for the purpose at hand, the recorded information can provide a precious source of examples from which to learn. (The importance of this kind of knowledge has been pointed out, for example, in [Malone et al. 1993].)


Project Leader
from RMM: receive(state, outcomes)
state= OK
outcomes = changes
to PM-AA: send(all modules approved)
synch(SD, plan from outcomes)
impossible changes
possible changes
RECOVERY PROTOCOL WITH RAC, TC AND PL
to PM-AA: send(state of involved modules suspended)
from Collector: receive(report-request)
synch(SD, report plan)
to PM-AA, send(state of involved modules under revision)
to Collector: send(final report)
synch(SD and JD, report)
Senior Designer
synch(PL, plan from outcomes)
to PM-AA: read(suspended module)
synch(PL, report plan)
synch(PL & JD, report)
REVISE SPECS
to PM-AA: send(new specs..., new designers)
CALL(....)
from PM-AA: receive(state of modules suspended)
Junior Designer
CALL(....)
from PM-AA: receive(new designer)
from PM-AA: receive(new specs)
CALL(....)
synch(PL and SD, report)

Project Management Artefact PM-AA

CONTENTS:
(For each module)
ID, state, related to "other modules ID", involved designers, specification, priority, deadline

VISIBILITY: (for all projects)
Project Leader:
all attributes: read;
state: update;
Senior Designer:
state: read only;
other attributes: read and update;
Junior Designer:
all attributes: read only.

UPDATE/READ REQUESTS:
{After its initialization, the AA is ready to receive from roles messages containing requests that respect the access rights specified above}

COORDINATION:
receive from PL (update(new state)) AND new state = suspended ---> send to involved Designers (new state);
receive from SD(update(new specification))
---> send to involved Designers (new specs)
AWARENESS:
receive from PL (update(new state)) AND new state = under discusssion
---> aw-send to involved Designers (new state).

Figure 4: The C^{2}M PROJECT MANAGEMENT. Gray transitions represent interactions.

3.3 Using a Computational Coordination Mechanism

The execution of aC^{2}M requires an additional effort in order to complete the partial specification provided at definition time. This effort is mainly in charge of the actor(s) assigned to the role responsible for this C^{2}M: to this aim they use the primitive *enact-instance*(X, Y), where X is the C^{2}M name and Y is the instance identifier. Obviously, the amount and type of additional information depend on the existing specification, on the time of activation, on the contingent situation. Here below some examples are sketched in the hypothesis that each additional piece of information can be added at any time with the only constraint that it is available when it becomes mandatory for the execution of theC^{2}M.

The person responsible for a C^{2}M, putting it at work for the first time, has to define which actors will play the roles mentioned in the C^{2}M. For example, the Project Leader can define her team by selecting which actors will play the roles of Senior and Junior Designers, and maintain it, by default, for all the subsequent activations of the Project Management C^{2}M. Obviously, at each instantiation the team can be modified, within the constraints stated by the role assignment policies. For example, the person responsible can delegate some of her duties to other actors after a negotiation with them and/or an authorization by some role playing as supervisor.

The assignment of roles to actors makes real activation possible: basically, this act determines the start of the transition from the nominal to the actual status of the C^{2}M. The activation can be done either under system control, when it depends on some external event: for example, a specific event signaled by the clock, or an incoming communication which plays the role of trigger of the protocol constituting the C^{2}M; or under actor's control by means of the primitive *activate*(Y). During execution of the C^{2}M, the Ariadne environment asks the actors to provide the missing information: for example, a `task` can be accomplished by executing some `activities`. The partial specification can contain some activities from which the actor can select the most appropriate one for her current needs; when no activity is mentioned, the actor can define it, on the fly, or be satisfied with a weaker support by the environment. In fact, in this case the latter opens a working space where the actor is totally in charge of the control. A similar reasoning can be done for all the attributes characterizing a `task`.

The space of possibility for incremental design is very rich, and the above examples provide only a taste of what the Ariadne environment aims to support. The main point we want to make is that this environment, thanks to the flexibility with which it can be used, supports not just the execution of a work process but also the definition of many ways to realize it and the selection of the coordination support it requires in the current situation. As mentioned in the introduction, each actor, each process, each instantiation can pose different demands. The selection process allows the actors to range from a conservative approach which relies on some existing effort and experience, typical of the unskilled/inexperienced/ill-motivated actors, to the most creative one, typical of the most innovative and self-confident members of the organization.

Since work process definition and selection can be conceived of as cooperative efforts supported by specific coordination mechanisms (e.g., the already mentioned conversation patterns for negotiating them), they can become a means for the propagation of experience and skill. Moreover, the environment makes it possible to record some definition and selection which resulted particularly effective and thus worthy of becoming a shared patrimony. This is achieved through the primitive *MakePermanent*(Y, new-name) which transforms an instance Y into a permanent C^2M that will be referenced by the new-name and then accessed through the primitives mentioned in the previous section, for design, learning and training purposes. A similar functionality is provided in, e.g., EGRET [Johnson 1992].

4 Modifying a Computational Coordination Mechanism

Simply selecting from among a set of possibilities or inventing a new C^2M or some of its components is not enough for managing all demands. We already mentioned the use of the repository of C^2Ms for reuse and adaptation. This opens the fundamental topic of how to support the modifications of existing C^2Ms as well as of existing grammars.

The modifications can pertain to each of the three levels of Ariadne's environment.
First of all, at the γ level, a grammar can be modified in some of its components through the primitive *modify*(G,modification-type), where G is a grammar identifier. In this case, the type of modifications can be in relation to either alternative CAWs or alternative Formal Relations. For example, *modify*(CONV-GR, structure) allows the user to change the structure used in the grammar: Labelled-SA-nets whose transitions are labelled in the Interactions could be substituted for the Labelled Graph, in order to allow for the definition of multi-party distributed conversations in addition to the standard dialogue patterns.

The modification could act also on the labelling functions by modifying the type of the objects constituting the labelling sets. For example, in the WF_GR we can allow only tasks as labels of the transitions and change the names of the protocol from Roles to generic labels. It is possible to modify also the semantics of a component of the grammar and not the component itself. For example, in the case of relational structures modeling concurrency one could select one of the alternative associated semantics.

As mentioned in [Section 2.3], the γ level is the domain of the so called 'application engineers' who have the skill for dealing with the formal aspects of the notation and for defining suitable combinations of the various Basic Elements to be used to construct a class of C^2Ms.

At the β level, Ariadne provides its users with the possibility of specifying in the attribute `adapted by` who has the right/duty to modify each specific component and/or their aggregation up to a whole C^2M, of the work process description. The value of this attribute can be either a `role` or a whole C^{2}M: this is a way to account for the recursive nature of articulation work [Schmidt and Simone 1996] since the component under modification becomes the Field of Work of a C^{2}M (as defined in [Fig. 1]) governing the modification process. Notice that people responsible and the modalities for the modifications can be totally different from the people responsible and the modalities for the definition of C^{2}M. This holds in general for all components of the C^{2}M. The environment provides the authorized actors with the primitive *modify-C^{2}M*(X, modification-type) where X is a C^{2}M name.

The question about modifications of a C^{2}M, as well as its definition, is not just about what is syntactically possible; rather, it is also about the correctness of the new description and of its relations with the descriptions of the work processes interacting with it. The environment provides the *define* and *modify* primitives with (interfaces to) tools for verifying the correctness of the new C^{2}M by exploiting checking techniques based on the Formal Relations contained in the notation, since the latter have an internal representation in terms of Petri Nets. This idea has a long tradition: it was present in DOMINO [Kreifelts et al. 1991a] where Petri-nets algorithms are exploited [Brauer et al. 1987]; in the proposal contained in [Ellis and Wainer 1994] where proving correctness of a modification of ICN [Ellis and Nutt 1980] exploits the theory of graph grammars [Ehrig et al. 1983] as ICNs are based on AND/OR-Graphs; and finally, in more recent proposals collected in [De Michelis and Pareschi 1996].

The identification of a tool supporting the check of correctness with other interacting processes introduces the crucial problem of how to manage the *propagation of changes*: this latter can be realized in different ways, depending on the type of modification. A first step is to evaluate whether or not the modifications have any impact on other C^{2}Ms.

This point opens one of the most challenging research areas to the improvement of adaptability in Ariadne, as it constitutes one of the main weakness of nearly all proposals. The modularity and focus on communication characterizing Ariadne suggest we look for support in verifying the consistency of the work process from the communication point of view. Although it is not integrated in Ariadne's framework, we worked on a tool for the semi-automatic verification of observational equivalence ([Milner 1980], [Pomello et al. 1992]) in concurrent systems. We can illustrate the idea in the scenario. If for some reason, the protocol of the Project Management C^{2}M is modified in terms of the involved `roles`, structure of `artifact` and assignment of `tasks` to `roles`, then the only constraint is the 'communication interface' with its environment, namely the communication with the Review Meeting C^{2}M (not discussed here and described in [Divitini and Simone 1996]. If all the `tasks/interactions` not involving this communication are considered as unobservable, then the interface just specifies the communicative events with the environment and their causal relation. Once in the new solution the internal behavior and communication are again made unobservable, then the modification is not affecting the other C^{2}Ms if the new interface is equivalent to the old one. That is, the communicative events with the environment satisfy the same causal relation.

The illustrated case is trivial since most of the communication is internal and therefore unobservable. The general case can be quite complex: our claim is that a computer support interacting with the designer who decides the correspondence of the old communicative events with the new ones could provide an invaluable support that can be applied uniformly to all components of a C^2M.

If the modification does not have any impact on the other C^2Ms then propagation of changes involves how to deal with the active instances of the modified C^2M: there are two possibilities here. An easy solution is to let them terminate according to the old definition and to consider the new one just when new instances are activated. If this is not the case, then the person responsible mentioned in the policy associated to the attribute `defined/modified by` has to deal with the definition of a new state from which to start the execution of the new C^2M. We will come back to this point later on.

If the modification does have an impact on other C^2Ms then propagation of changes is dramatically more complex. In fact, beside the problem mentioned above, propagation of changes has to deal with the alignment of the behavior of all the involved C^2Ms, possibly at both instantiation and definition levels. If the impacts can be filtered by the modified C^2M interface, then the problem is just to act on this interface so that the new C^2M appears unmodified to the external world: this case was discussed in the scenario. Otherwise, all the responsible roles have to be involved for realizing the alignment. Ariadne does not provide any 'panacea' to this problem: rather it makes the responsible roles visible, supports the cooperative redefinition by means of an ad hoc C^2M (specified in the policy associated to the attributes `defined/modified by),` and in principle could support the evaluation of the external impacts just applying the technique based on the above mentioned observation equivalence.

Let's go back to the scenario. In [Section 3.1] we said that when the organization was established the management of a research area was seen simply as a juxtaposition of the management of the single projects belonging to the area. The Research Area Management C^2M can therefore be described in the following way:

Research Area Management = Project Management-1 // .... // Project Management-k

where *Project Management n* is the mechanism whose construction has been illustrated in [Section 3.2]. These mechanisms are fully independent.

In a subsequent stage, the two considered C^2Ms had to be modified as the outcome of an evolution of the organization demands. As the complexity of the projects increased the RAC responsibilities became more operative, in particular in the attempt of guaranteeing a uniform quality (and timely delivering) of the reports submitted to the periodic review meetings where the organization monitors the status of the on-going projects. Then, the Research Area Coordinator (RAC) became not just a reference in case of exceptional situations (namely, the handling of impossible changes, as illustrated in [Fig. 4]); rather, it had to play an active role in guaranteeing the quality of the process by synchronizing its behavior with that of the people involved in each project management, for fully anticipated tasks. In this view, the Research Area Management C^2M was no longer to be considered as a set of juxtaposed Project Management C^2Ms, but as a structure of independent mechanisms that have to synchronize at some points through the communication with the `role` responsible for the structure. The new C^2M, still called Research Area Management, 'contains' the others in the sense illustrated in [Fig. 5]. An artifact is introduced: its structure was defined according the new responsibilities of the RAC. We are not giving here its detailed components: for our purposes it is just worth mentioning that it contains the references to the Project Management C^2Ms. In our experience, this is a quite recurrent situation that arises whenever there is the need to harmonize the parallel execution of different instances of a same mechanism. This is why we think that this situation could be defined as a predefined pattern of information structure and communication capabilities to be provided by Ariadne and specialized by the designers.

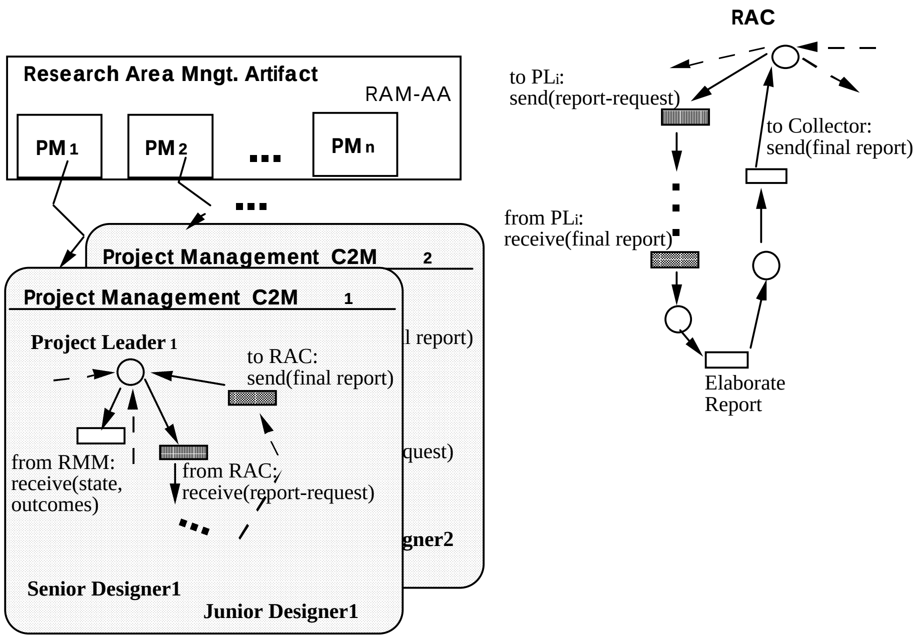

RAC
Research Area Mngt. Artifact
RAM-AA
PM 1
PM 2
...
PM n
Project Management C2M 2
Project Management C2M 1
Project Leader 1
to RAC: send(final report)
from RMM: receive(state, outcomes)
from RAC: receive(report-request)
l report)
quest)
gner2
Senior Designer1
Junior Designer1
to PLi: send(report-request)
to Collector: send(final report)
from PLi: receive(final report)
Elaborate Report

Figure 5: The new C^2M RESEARCH AREA MANAGEMENT

The (part of the) protocol governing the new RAC behavior explicitly states a communication with the various Project Leaders for preparing the reports. This has an impact on each Project Management C^2M (within the Research Area Management mechanism): the only required modification is the change of source and destination of the communication about the report preparation. In fact, what before had to be communicated to the role responsible for collecting the reports (the Collector mentioned in the `interactions` of the Project Management mechanism) is now communicated to the Research Area Coordinator. This modification should have some effect on the C^2M supporting the review meeting (again, not considered here) because it implies the modification of one of its role. The decision about how to realize this modification was the outcome of a discussion among the people in charge of the reorganization: the main concern was about how to manage the related impacts. Two strategies were considered. The modification is to be considered either as a re-organization internal to the Research Area Management process or as part of a more global reorganization, as a 'must' coming from outside the process itself. In the former case, the Collector remains fully unaware of this modification. Collector sends out report-requests and waits for the reports from the Project Leaders. From the technical point of view, the interface managing the external communication of Research Area Management C^2M has to take care of the 'translation' of these addresses. In the second case, if the modification has to be part of an explicit re-organization, then Collector in the Review Meeting process has to be explicitly involved too: that is, here the communication with the Project Leaders has to be changed accordingly. Various factors, among which the small size of the organization, led to the selection of this second possibility.

If we abstract from the specific scenario and consider a more complex situation, then the people in charge of the reorganization could adopt a strategy which is a combination of the two in order to better manage the propagation of changes. In fact, considering in the first place a modification as an event local to a process (C^2M) allows us to test its consequences locally before making the modification a public event explicitly involving other processes. When the solution is satisfactory, then the other processes are modified with a greater level of confidence. And the process originating the modification will remain stable for a while.

This way of proceeding can have a positive impact in supporting the active involvement of everyone in modifying work processes, avoiding the delegation of all changes to the management. In a highly dynamic working environment, in fact, the modularity of Ariadne allows experimentation of new solutions locally, so that the lack of impact on the external world can increase the creativity of the people directly involved in the process execution. In this way, changes are the result of experiences gained "on the field": people learn to work together and adapt their interaction in order both to meet changing demands in the environment and to perform more effectively. This is not to deny the importance, within the organization, of more global changes that can be addressed only by people with an overall vision of the organization and the evolving market requirements. A system like Ariadne, we believe, can provide a smooth integration between changes coming from the bottom, and therefore "situated" in a specific working setting, and ones coming from the top, and therefore "enlighten" by a global vision of organizational needs. We want to underline that this is a possibility provided by Ariadne, the extent to which this possibility is actualized within the specific organization depends on the policies defined within the organization itself.

Finally, at the α level, the modifications concern temporary changes of an activated mechanism in face of some unexpected event or of events whose handling strictly depends on the current situation. Two are the primitives available to this end.

The primitive *modify-instance*(X) allows for structural modifications of the instance: for example, if its source C^2M exploits graphs as a formal structure, *modify-instance*(X) allows the insertion/deletion of arcs and nodes and/or the change of their labelling functions. These modifications affect just the current instance while the source C^2M remains unchanged. As for the case of the permanent changes at the β level, the primitive *modify* can interact with tools for the verification of the consistency of the modified instance. Of course, if the changes have an impact outside it, then the alignment to the new configuration has to be negotiated among the interested actors, possibly by means of a C^2M supporting negotiation. The decided alignment can be 'enforced' by the following primitive.

The primitive *enforce*(X, new-configuration) makes an instance to proceed from a new configuration with respect to the current one. For example, if we consider a graph-based formalism, the configuration is made of the current node; in the case of Petri-nets based formalisms, the configurations are the markings; in the case of an active artifact the configuration is the current set of values. Then, the primitive allows us to change node, marking and values, respectively, in a way that is independent of the previous configurations. As discussed in [Ellis et al. 1995], enforcing a new configuration poses serious consistency problems and requires ad hoc tools for guaranteeing a consistent future behavior. Beside the techniques they propose for Petri nets-based descriptions, we would like to mention the possibility of using the notion of state equivalence of Petri nets [Bernardinello et al. 1996], a notion allowing two net systems to be compared through suitable morphisms relating their structure and their state spaces. In this way, a designer can check if the new state leads to a configuration from which an anticipated behavior can start again.

The already mentioned primitive *MakePermanent*(X, new-name) which allows us to transform an instance into a permanent C^2M that will be referenced by the new-name can be used after the activation of the primitive allowing for structural modifications that are becoming recurrent so that they can be made permanently available for future uses.

In the scenario, an example of modification of the current state is generated by the `task` Recovery Protocol in the Project Management C^2M: this task is activated when the review process requires changes of the project that are considered as impossible in the current situation. In this case, there are two possible outcomes: either the project is forced to a final state or all the current activities are suspended until a negotiation of those changes leads to another set of requests. These two situations do not create serious consistency problems because the enforced states belong to the space of possible states of the current description. In fact, a project can terminate successfully, and its modules are suspended during the revision of their specifications by the Senior Designer.

As a concluding remark, we can say that if we observe an organization even for a short period of time, it is possible to witness numerous kinds of changes, going from small adjustments to major changes that make the processes more suitable to the dynamic needs of the evolving organization. Even if we consider only the few examples provided above, it is possible to note that some modifications have a strong impact on the way processes are defined and on how people work, while others are more circumscribed and their effect can be limited to a single execution of a process. It is clear that different modifications require that the user performing them possesses different competencies and a different authority within the organization. It is not possible to define general rules, because this depends on the policies applied by the organization as well as by the specific process or task under concern (e.g., organize a meeting vs. managing a nuclear plant). Ariadne provides a framework where policies can be defined at what is dynamically considered as the right level of granularity. Moreover, the level of control that is necessary, above all in terms of authorized changes, is strictly related to the skill of the user under concern and her role within the organization. So, for example, newcomers might not be allowed to modify the way they act in a cooperative effort, because they may not be aware of the consequences that this can have on the overall process, while this restriction can be relaxed when the user gains more experience.

In combining the knowledge and the policies collected in the specific (component of a) C^{2}M, in the role that is responsible for the modifications and the actor that is playing it in a contingent situation, the system can provide a high degree of flexibility in the description of the different constraints that influence the expected behavior of the members of a group and their degree of freedom in acting within the organization. Moreover, the system can use all these pieces of knowledge to tailor its interaction to the specific user in question. For example, it is possible to specify that a certain role can modify a specific component only if authorized by a supervisor whenever the actor playing it has no specific skill, the authorization can be weakened to just an acknowledgment as the experience of the user increases to leave place to a total freedom for the more experienced ones. The system can act differently in the different situation by activating automatically a devoted mechanism, a remainder or simply non-acting when this is not necessary. Ariadne therefore provides its user with the possibility of defining different policies and relaxing them on the basis of different external constraints (like, for example, the profile of a specific user).

5 ABACO: an Agent Based Implementation of Ariadne

Ariadne allows the construction of a C^{2}M 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.

In ABACO agents are characterized following a behavioral criterion, as described in [Genesereth and Ketchpel 1994]: software agents are defined as "components that communicate with their peers by exchanging messages in an expressive agent communication language". Stated in another way: "An entity is a software agent if and only if it communicates correctly in an agent communication language.... This means that the entity must be able to read and write these messages and that the entity must abide by the behavioral constraints implicit in the meanings of those messages." This criterion does not consider other aspects (for example, social ability, pro-activeness, autonomy) and we don't mean to underestimate the importance of the agent internal architecture. In our perspective, the behavioral criterion was an essential starting point for defining the macro architecture of the system. Different actual implementations of ABACO can exploit different internal structures, possibly using heterogeneous agents specialized in relation to the services that they provide, but preserving the basic communicative behavior that will be described for each type of agent. In the following, we will present the architecture of ABACO and its agent communication language.

5.1 The Multi-Layer Structure of ABACO

In very general terms, each component of Ariadne, from the basic elements up to the composite C^{2}Ms obtained from the composition of more elemental C^{2}Ms, is realized as an agent belonging to a specific type. Each type is characterized by its capability of communicating with the other agents that dynamically constitute the environment.
Following the idea of realizing each element of Ariadne as an agent, ABACO exhibits a multi-layer architecture since C^{2}Ms are compound entities that are built on top of basic elements. ABACO is organized in three layers [Fig.6].
Since the Categories of Articulation Work (CAW) are the building blocks of C^{2}Ms made available by Ariadne, the agents realizing these categories populate the *first layer* of the architecture. (For the sake of simplicity, agents will be denoted, when this does not create ambiguity, by the name of the corresponding concept.) Each CAW agent manages a set of information that varies from one category to another, according to the attributes that characterize the category in the underlying model of articulation work sketched in [Section 2.2]. These agents can provide the managed information to other agents whenever it is needed and are characterized by a communicative behavior that makes them able to interact with other agents and with the Organizational Context where they are defined and adapted. Specifically, each CAW agent is characterized by the possibility of making the environment *aware* of its internal conditions (as explained in [Section 2.4]). The structure of the CAWs represents a common ontology to which all the agents of ABACO subscribe.

The *second layer* is related to single C^{2}Ms. From the behavior point of view, a C^{2}M is the parallel composition of the behaviors of the related CAWs, Active Artifact and Protocol on the basis of their communication capabilities. As explained in [Section 2.4], the artifact assumes an essential and active role in mediating the articulation work among the cooperating entities involved in the C^{2}M by notifying them of appropriate information in presence of particular conditions and by actively participating in the articulation work effort thanks to coordination capabilities. From the architectural point of view, an AA is an agent specialized in the management of the related information and is able to communicate with its environment in order either to notify appropriate information in presence of defined conditions (awareness) or to collect from agents and convey to agents information which is compulsory to coordinate their work (coordination). Conventions and procedures are represented in Ariadne through the notion of Protocol. A protocol is a compound entity obtained from the composition of CAWs as in a cooperative arrangement conventions and procedures can be expressed in terms of relations among Tasks, Roles, Actors, Actions, Interactions and Resources. >From the architectural point of view, protocols correspond to agents that, among other, control the activation of other CAW agents, as specified by the underlying Formal Relation.

Finally, the *third layer* is populated by the agents obtained from the composition of already existing C^2Ms. The implementation of the composition of C^2Ms is based on the definition of an Interface agent specialized in managing external communication. The Interface agent plays the role of both facilitator and monitor of the embedded C^2M in order to handle the additional communication needs derived from the composition of different C^2Ms [Genesereth and Ketchpel 1994]. The interface, for example, establishes which internal information can be mutually accessed by and communicated to other mechanisms and controls the access to the artifact from the external world (i.e., agents that do not belong to the mechanism of which the artifact is part). Moreover, the interface monitors the behavior of the C^2M to handle the additional communication needs derived from co-existence of different C^2Ms. Finally, the interface is the place where mechanisms can be made tolerant to the modifications of their world. In fact, interfaces can be conceived of as agents specialized not only in the management of the communication but also in the 'translation' or redirection of the incoming/outcoming information in a format that can be properly interpreted. If the translation is not possible, then the interfaces will notify it and support the activation of the suitable negotiation/recovery protocols.

Notice that the Interface agent can be employed in order to establish the type of interoperability between a C^2M and the field of work whose activities the C^2M is articulating.

5.2 The Interoperability Language

According to their definition, agents share the same communication language. An essential step in the development of ABACO was, therefore, the definition of the language that the agents use to communicate, that we call Interoperability Language (IL). The basic primitives of the language were determined considering the types of communication each agent has to realize in order to interoperate with other agents. Three modes characterize the interactions among agents [3]:

In *subscription mode* an agent makes the behavior of another agent part of its own behavior. For example, a resource can be accessed in the subscription mode to

3 In an initial stage these modes were determined comparing various field studies in order to determine how C^2Ms interact. The three modes were then usefully exploited for describing the communication among ABACO agents.

activate the policies governing its usage. Another interesting example is when a protocol subscribes to other C^2M in order to support a negotiation or to activate a process that is coordinated by another C^2M, like when a protocol make reference to a C^2M supporting conversations among roles [Winograd and Flores 1986]. This C^2M is basically constituted by interactions combined by causal relations and by an active artifact describing the status and history of the conversation. Through the reference to these types of C^2Ms, the communication capability of Ariadne's components (and by consequence, of the Interoperability Language) is therefore very expressive and flexible in terms of patterns of interactions [Labrou and Finin 1994].

In *inscription mode* a C^2M provides information about its current state to another C^2M (or, conversely, a C^2M obtains information about the current state of another C^2M). The inscription can be done in two ways: the reaction by the target C^2M can be either *compulsory* or *voluntary*. In the case of compulsory inscription, the target C^2M is expected to react accordingly and, if it 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 C^2M. In the case of voluntary inscription, the reaction of the target C^2M 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 C^2M to voluntarily filter the provided information ([Malone et al. 1987], [Gasparotti and Simone 1990], **[**Fuchs et al. 1995]). The compulsory inscription mode typically expresses the reading from and writing to the artifact by the protocol in order to acquire and make visible imperative information. On the other hand, the voluntary inscription mode is typically used by the artifact to convey awareness of its internal changes to the other components of the mechanism.

In *prescription mode* a C^2M over-writes the definition of the target C^2M's behavior. This interaction mode allows Ariadne to honor the recursive nature of articulation work. In fact, in the prescription mode a given C^2M can change the definition of another C^2M, that is, the definition of its protocol, or its specification, for example, by enforcing a special state during its execution.

It is worth mentioning that the defined modes of interactions are used both at the level of the mechanism and its components (categories of articulation work, artifact, protocols) to express their interactions as well as in the formalization of the primitives of the notation [Divitini et al. 1996]. Then, the IL has been uniformly used in the architecture not only for expressing the communication made explicit by the designer of C^2Ms (for example, to describe the communication between two C^2Ms) but also to 'implement' the implicit (that is, system-defined) communication among all the elements of the notation (for example, when a CAW makes reference to another CAW in its attributes). In this way, the Interoperability Language is the basic means for realizing 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. This basic property led us to avoid using a standard Agent Communication Language (e.g., KQML [Genesereth and Ketchpel 1994], [Labrou and Finin 1994]) and to postpone

the possible mapping of IL into a standard when ABACO will be implemented in all its aspects.
ABACO has been designed so that its elements can be composed in a flexible way in order to obtain the desired expressive power to define, adapt and link C^2Ms.
The primitives that have been presented in the previous sections are all realized as services provided by the single agents. All the agents, for example, can deal with messages in a prescription mode that require for the modification of their internal structure or communicative behavior. Messages are processed by agents accordingly to locally defined knowledge. Before providing the required service, the agent verifies that it comes from an authorized source and it applies the required policies (for example, before starting a modification coming from X, the agent has to require an additional authorization to a supervisor).

A first demonstrator of the ideas presented above has been realized in order to test the Interoperability Language. Another demonstrator is currently under development in the Java environment. Here, the main effort is devoted to provide the agents of ABACO with reflective capabilities. In fact, the agents of ABACO act in a highly dynamic environment: (the components of) a C^2M can be changed at any moment in order to deal with local contingencies or permanent changes in organization; at the same time, new C^2Ms can be introduced in order to deal with different aspects of cooperative activities. In the present architecture, modifications and the subsequent propagation of changes are mainly under user control, with a little help from the system. Without entering into specific details, we claim that the relevant point here is that ABACO has to be equipped with reflective features ([Maes 1988], [Yonezawa 1990]) in order to provide a stronger support to the management of the modifications as well as to the automatic propagation of changes implied by the adopted strategy.

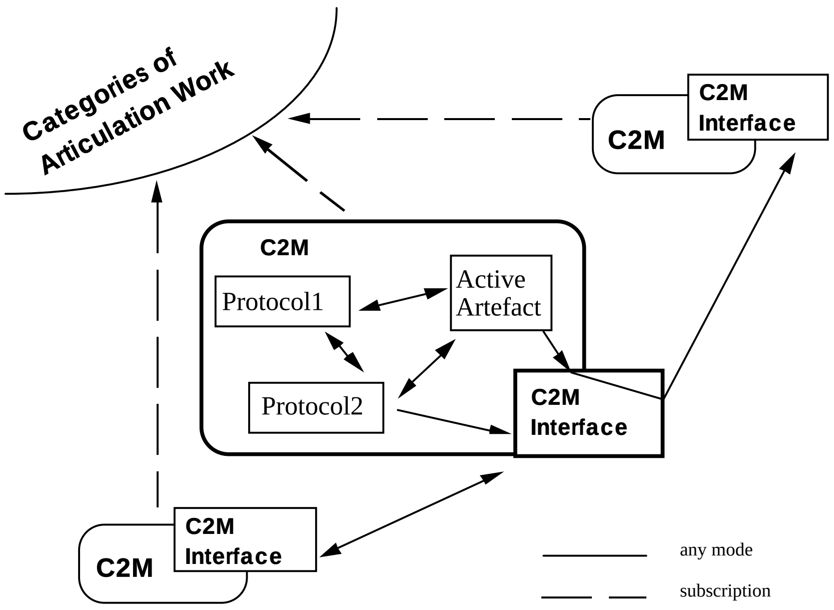


Figure 6: The structure of ABACO

6 Conclusions

The paper has presented an approach to the construction of flexible coordination mechanisms supporting the articulation work of actors involved in cooperative activities. Flexibility is achieved by combining linguistic, functional and architectural features. Moreover, flexibility is presented as one of the basic requirements that make the workflow technology both a support to coordination and an enabler of the creative management and learning of the knowledge on work processes.
Different improvements of the approach are currently under consideration.

First, the way in which communication is described (and implemented in ABACO) allows a direct integration of functionalities supporting the contexts in which communication occurs [Divitini and Simone 1994]. More work will be devoted to User Interface agents (UI). This agent, not mentioned in this paper, is devoted to the interaction of the system with the user, taking into account that they are working in cooperative settings and in multiple contexts. UI agents can naturally incorporate some standard services as filtering [Sheth and Maes 1993] and intelligent assistance [Greif 1994] based on appropriate User Models [Kobsa and Wahlster 1989] and on users preferences. Moreover, providing a well established model of articulation work, the proposed framework can be the basis for the definition of the notion of Group Models and Profiles.

Second, ABACO is conceived as the first step towards an agent based software infrastructure intended to support the prototyping of CSCW applications based on the concept of Computational Coordination Mechanisms. In the presented work we have defined the architecture of the system, i.e., its component-agents and their relationships. Moreover, the main services that must be provided by each agent have been identified. The use of a more complex internal architecture could support, for example, a proactive behavior of the agents. More research in this direction is required.

Finally, more work is required in order to study the possibility of linking a mechanism to service agents or to legacy software. A C^2M system must be able to interoperate with existing applications; moreover, specific C^2M systems can be built whenever the use of an application in a cooperative environment requires coordination of the involved actors. The agent based approach seems to facilitate the solution of these problems. In fact, interoperability with legacy applications is a well recognized problem in the area of agent based programming and some solutions have already been proposed [Genesereth and Ketchpel 1994].

References

[Bernardinello and De Cindio 1992] Bernardinello, L., De Cindio, F.: "A Survey of Basic Net Models and Modular Net Classes"; Advances in Petri Nets 92, LNCS 609, Springer-Verlag, Berlin, Germany (1992), 304-351.

[Bernardinello et al. 1996] Bernardinello, L., Pomello, L., Simone, C.: "A class of morphisms for the refinement of EN systems"; DSI-Milano/Technical Report n. 181/96, Italy (1996), submitted.

[Bowers et al. 1995] Bowers, J., Button, G., Sharrock, W.: "Workflow from Within and Without: Technology and Cooperative Work on the Print Industry Shopfloor"; Proc. of ECSCW'95 (Fourth European Conference on Computer-Supported Cooperative Work), Kluwer Academic Publishers, Dordrecht, Germany (1995), 51-66.

[Brauer et al. 1987] Brauer, W., Reisig, W., Rozenberg, G. (eds.): "Petri Nets: Central Models and Their Properties"; LNCS 254, Springer-Verlag, Berlin, Germany (1987).

[De Cindio et al. 1982] De Cindio, F., De Michelis, G., Pomello, L., Simone, C.: "Superposed Automata Nets"; Application and Theory of Petri Nets, IFB 52, Springer-Verlag, Berlin, Germany (1982).

[De Michelis and Pareschi 1996] De Michelis, G., Pareschi, R. (eds.): "Workshop on Adaptive Workflow"; Proc. PAKM '96, Basel, Switzerland (1996).

[Divitini and Simone 1994] Divitini, M., Simone, C.: "A Prototype for Providing Users with the Contexts of Cooperation"; Proc. ECCE7, GMD, Bonn, Germany (1994), 253-270.

[Divitini and Simone 1996] Divitini, M., Simone, C.: "Ariadne: a framework to construct flexible workflow systems"; Proc. PAKM'96 (First International Conference on Practical

Aspects of Knowledge Management) - Workshop on Adaptive Workflow, Basel, Switzerland (1996).

[Divitini et al. 1996] Divitini, M., Simone, C., Schmidt, K.: "ABACO: Coordination Mechanisms in a Multi-agent Perspective"; Proc. of the 2nd International Conference on the Design of Cooperative Systems, INRIA, France (1996), 103-122.

[Dourish et al. 1996] Dourish, P., Holmes, J., MacLean, A., Marqvardsen, P., Zbyslaw, A.: "Freeflow: mediating between representation and action in workflow systems"; Proc. CSCW'96 (International Conference on Computer Supported Cooperative Work),. ACM Press, New York (1996), 190-198.

[Ehrig et al. 1983] Ehrig, H., Nagl, M., Rozenberg, G. (eds.): "Graph-Grammars and Their Application to Computer Science (2nd International Workshop*)"*; Lecture Notes in Computer Science, Springer-Verlag, Berlin, Germany (1983).

[Ellis and Nutt 1980] Ellis, C., Nutt, G.: "Office information systems and computer science"; ACM Computing Surveys, 12, 1 (1988), 27-60.

[Ellis et al. 1995] Ellis, C.A., Keddara, K., Rozenberg G.: "Dynamic Change within Workflow Systems"; Proc. of COOCS'95, ACM Press, New York (1995), 10-21.

[Ellis and Wainer 1994] Ellis, C. A., Wainer, J.: "Goal-based Models of Collaboration"; Collaborative Computing, 1, 1 (1994), 61-86.

[Fitzpatrick et al. 1996] Fitzpatrick, G., Kaplan, S., Mansfield, T.: "Physical spaces, virtual places and social worlds: a study of the work in the virtual"; Proc. of CSCW'96 (Computer Supported Cooperative Work), ACM Press, New York (1996), 334-343.

[Fuchs et al. 1995] Fuchs, L., Pankoke-Babbatz, U., Prinz, W.: "Supporting cooperative awareness with local event mechanisms: the Groupdesk system"; Proc. ECSCW'95 (4th European Conference on Computer Supported Cooperative Work), Kluwer Academic Publishers, Dordrecht, Germany (1995), 247-262.

[Gasparotti and Simone 1990] Gasparotti, P., Simone, C.: "A user defined environment for handling conversations"; Proc. of IFIP WG8.4 Conference on Multi-User Interfaces and Applications (1990), 271-289.

[Genesereth and Ketchpel 1994] Genesereth, M.R., Ketchpel S.P.: "Software Agents"; Communication of the ACM, 37, 7 (1994), 48-54.

[Glance et al. 1996] Glance, N. S., Pagani, D.S., Pareschi, R.: "Generalized Process Structure Grammars (GSPS) for flexible representation of work"; Proc. CSCW'96 (International Conference on Computer Supported Cooperative Work), ACM Press, New York (1996), 180-189.

[Greif 1994] Greif, I.: "Desktop Agents in Group-Enabled Products"; Communications of the ACM, 37, 7 (1994), 100-105.

[Johnson 1992] Johnson, P.: " Supporting Exploratory CSCW with the EGRET Framework"; Proc. CSCW’92 (Conference on Computer-Supported Cooperative Work), ACM Press, New York (1992), 298-305.

[Kaplan et al. 1992] Kaplan, S. M., Tolone, W.J., Bogia, D.P., Bignoli, C.: "Flexible, Active Support for Collaborative Work with Conversation Builder"; Proc. CSCW’92 (Conference on Computer-Supported Cooperative Work), ACM Press, New York (1992), 378-385.

[Kobsa and Wahlster 1989] Kobsa, A., Wahlster, W. (eds.): "User models in dialog systems"; Springer-Verlag, Berlin/ Heidelberg, Germany (1989).

[Kreifelts et al. 1991a] Kreifelts, T, Hinrichs, H., Klein, K.H., Seuffert, P., Woetzel, G.: "Experiences with the DOMINO Office Procedure System"; Proc. ECSCW ’91 (Second European Conference on Computer-Supported Cooperative Work), Kluwer Academic Publishers, Amsterdam (1991), 117-130.

[Kreifelts et al. 1991b] Kreifelts, T, Victor, F., Woetzel, G., Woitass, M.: "Supporting the design of office procedures in the DOMINO system"; Studies in Computer Supported Cooperative Work. Theory, Practice and Design,. North-Holland, Amsterdam etc. (1991), 131-144.

[Kreifelts et al. 1993] Kreifelts, T., Hinrichs, H., Woetzel, G.: "Sharing To-Do Lists with a Distributed Task Manager"; Proc. ECSCW’93 (Third European Conference on Computer-Supported Cooperative Work), Kluwer Academic Publishers, Dordrecht, Germany (1993), 31-46.

[Labrou and Finin 1994] Labrou, Y., Finin, T.: "A semantics approach for KQML - a general purpose communication language for software agents"; Proc. CIKM'94 (3rd International Conference on Information and Knowledge Management), (1994). Available as http://www.cs.umbc.edu/kqml/papers/kqml-semantics.ps.

[MacLean et al. 1990] MacLean, A., Carter, K., Lövstarnd, L., Moran, T.: "User-tailorable systems: Pressing the issue with Buttons"; Proc.CHI'90, ACM Press, New York (1990).

[Maes 1988] Maes, P.: "Computational Reflection"; The Knowledge Engineering Review, (1988), 11-19.

[Malone et al. 1993] Malone, T. W., Crowston, K., Lee J., Pentland, B.: "Tools for inventing organizations: Toward a handbook of organizational processes"; Proc. of the 2nd IEEE Workshop on Enabling Technologies Infrastructure for Collaborative Enterprises, Morgantown, (1993).

[Malone et al. 1987] Malone, T. W., Grant, K. R., Lai, K. -Y., Rao, R., Rosenblitt, D.: "Semistructured messages are surprisingly useful for computer-supported coordination"; ACM Transactions on Office Information Systems, 5, 2 (1987), 115-131.

[Malone et al. 1992] Malone, T. W., Lai, K. -Y., Fry, C.: "Experiments with Oval: A Radically Tailorable Tool for Cooperative Work"; Proc. CSCW'92 (Conference on Computer-Supported Cooperative Work), ACM Press, New York (1992), 289-297.

[Milner 1980] Milner, R.: "A Calculus of Communicating Systems"; Lecture Notes in Computer Science, LNCS 92, Springer-Verlag, Berlin, Germany (1980).

[Pomello et al. 1992] Pomello, L., Rozenberg, G., Simone, C.: "A Survey of Equivalence Notions for Petri Net based Systems"; Advances in Petri Nets 92, LNCS 609, Springer-Verlag, Berlin, Germany (1992),410-472.

[Schmidt 1997] Schmidt, K.: "Of Maps and Scripts - The status of formal constructs in cooperative work"; Risoe Technical Report, Roskilde, Denmark (1997), submitted.

[Schmidt and Simone 1996] Schmidt, K., Simone, C.: "Coordination Mechanisms: Towards a conceptual foundation for CSCW systems design"; Computer Supported Cooperative Work- An International Journal, 5, 2-3 (1996), 155-200.

[Shepherd et al. 1990] Shepherd, A., Mayer, N., Kuchinsky, A.: "Strudel -- An extensible electronic conversation toolkit"; Proc. of CSCW'90 (Conference on Computer Supported Cooperative Work), ACM Press, New York (1990), 93-104.

[Sheth and Maes 1993] Sheth, B., Maes, P.: "Evolving agents for personalized information filtering"; Proc. of the Ninth Conference On Artificial Intelligence for Applications, IEEE Computer Society Press (1993), 1-11.

[Simone et al. 1995] Simone, C., Divitini, M., Schmidt, K.: "A notation for malleable and interoperable coordination mechanisms for CSCW systems"; Proc. of COOCS'95, ACM Press, New York (1995), 44-54.

[Swenson et al. 1994] Swenson, K. D., Maxwell, R.J., Matsumoto, T., Saghari, T., Irwin, K.: "A Business Process Environment Supporting Collaborative Planning"; Collaborative Computing, 1, 1 (1994), 15-34.

[Trevor et al. 1993] Trevor, J., Rodden, T., Blair, G.: "COLA: a Lightweight Platform for CSCW"; Proc. ECSCW ’93 (Third European Conference on Computer-Supported Cooperative Work), Kluwer Academic Publishers, Dordrecht (1993), 15-30.

[Winograd and Flores 1986] Winograd, T., Flores, F.: "Understanding Computers and Cognition: A New Foundation for Design"; Ablex Publishing Corp., Norwood, New Jersey (1986).

[Yonezawa 1990] Yonezawa, A. (ed.): "ABCL: An Object-Oriented Concurrent System"; MIT Press, Cambridge, MA (1990).