In: *Gert-Jan de Vreede, L. A. Guerrero, and G. M. Raventós (Eds.),* ***10th International Workshop on Groupware - CRIWG 2004,*** *San José, Costa Rica. Lecture Notes in Computer Science, Vol. 3198 – Groupware Design, Implementation, and Use, p.121-136. Springer-Verlag, 2004*

Combining Communication and Coordination toward Articulation of Collaborative Activities

Alberto Barbosa Raposo[1], Marco Aurélio Gerosa[2], Hugo Fuks[2]

[1] Computer Graphics Group (Tecgraf)
Computer Science Department – Catholic University of Rio de Janeiro (PUC-Rio)
R. Marquês de São Vicente, 225, Gávea, Rio de Janeiro – RJ, 22453-900, Brazil
abraposo@tecgraf.puc-rio.br
[2] Software Engineering Laboratory (LES)
Computer Science Department – Catholic University of Rio de Janeiro (PUC-Rio)
R. Marquês de São Vicente, 225, Gávea, Rio de Janeiro – RJ, 22453-900, Brazil
{gerosa,hugo}@inf.puc-rio.br

Abstract. In this paper, we present a proposal for the articulation of collaborative activities based on communication and coordination representation models. Articulation is essential in any kind of collaboration and involves pre-articulation of the tasks, their management, and post-articulation. For the representation of pre- and post-articulation phases, conversation clichés (communication) are used. For the coordination phase, a model separating the tasks and their interdependencies is used. The articulation schema, which is especially suited to e-business applications, is then applied to a business-web example.

1 Introduction

Planning is an essential activity to develop collaboratively any major task, since it ensures that the desired goal will result from individual tasks. In Computer Supported Cooperative Work (CSCW), the notion of planning is realized by articulation, a set of activities required to manage the distributed nature of collaborative work [18].

The articulation work involves the pre-articulation of the tasks, their management and post-articulation. Pre-articulation involves the actions that are necessary to prepare the collaboration, normally concluded before the collaborative work begins, like the identification of the objectives, the mapping out of these objectives into tasks, the selection of the participants, the distribution of tasks among them, etc. The post-articulation phase occurs after the end of the tasks and involves the evaluation and the analysis of the tasks that were carried out and the documentation of the collaborative process.

The management of the carrying out of the tasks, or coordination, is the part of the articulation work with a distinct, dynamic nature, needing to be renegotiated in an almost continuous fashion throughout the collaboration period. It can be defined as the act of managing interdependencies between tasks that are carried out to achieve an objective [13].

Articulation is essential to any kind of collaboration. In spite of that, coordination does not need to appear explicitly in some kinds of computer-supported collaborative activities–called loosely integrated collaborative activities [15]–such as those realized by means of chats or audio/videoconferences. These activities are deeply associated with social relations and generally are satisfactorily coordinated by the standing social protocol, which is characterized by the absence of any explicit coordination mechanism among the activities, trusting users' abilities to mediate interactions (the coordination is culturally established and strongly dependent on mutual awareness). By coordination mechanism, we mean "a specialized software device, which interacts with a specific software application so as to support articulation work" [19].

On the other hand, there is a large group of activities (tightly integrated collaborative activities) that require sophisticated coordination mechanisms in order to be efficiently supported by computer systems. In this kind of activity, tasks depend on one another to start, to be performed, and/or to end. Workflow management systems are examples of computational support suited for this kind of activity.

Business-webs (b-webs), which are partner networks of producers, suppliers, service providers, infrastructure companies, and customers linked via digital channels [23], is an example of process realized by tightly integrated collaborative activities, demanding computational support for coordination in the workplace. In this context, this paper presents an articulation schema based on conversation clichés and a coordination model separating the tasks and their interdependencies.

It is important to mention that this work follows a collaboration model based on Communication, Coordination and Cooperation (the 3C Model [6], [10]). According to this model, in order to collaborate, people communicate. During this communication, commitments may be generated. These commitments imply tasks that will be necessary to have the job done. These tasks are managed by coordination that organizes the group and guarantees that the tasks are accomplished in the correct order, at the correct time and according to the imposed restrictions. During cooperation, which is the joint operation in a shared space, the members of the group accomplish tasks through the interaction of the individuals and the artifacts of the work environment. However, while working, the necessities of renegotiating and of taking decisions appear. Then, a new round of communication is demanded, which in turn may modify and generate more commitments that will need coordination to reorganize the tasks that are performed during cooperation.

In the following section, the representation models are explained. Section 3 shows how those models may be applied to b-webs. In Section 4, the articulation schema combining both representation models is discussed in the light of related work.

2 Representation Models

This section presents two representation models embracing the whole spectrum of articulation. For the pre- and post-articulation phases, we propose the use of conversation clichés for negotiating, structuring and evaluating the collaborative work [12]. For the coordination phase, we propose a model where the commitments generated during the conversation (pre-articulation) define the collaborative activity, and coor-

dination mechanisms are created to manage the interdependencies between the tasks carried out by the members of the group [15].

2.1 Conversation Clichés

Commitments are the results of a conversation for action [25], representing an informal contract with other people involved in that conversation. They may indicate a new responsibility, an obligation, a restriction or a decision, guiding the people's actions [8].

There are other notions of commitment in the computing literature [2], [4], [11], [27]. Our notion of commitment could be seen as a proper subset of Bond's [2] concept of commitment. Although we agree on the behavioral aspects of commitment, we have different views about retracting commitments. Moreover, we provide a calculus for dealing with commitments [9].

As the commitments originate the group tasks, which will be managed by the coordination, the representation of commitments has an important role in the collaboration as a whole. However, since these commitments come from the natural language communication, it is difficult to capture them. In spite of that, some kinds of negotiation, such as those involving a well-known domain (such as an e-business setting), tend to be structured and repetitive. Therefore, it is possible to identify previously the parts of the discourse structure that are repetitive–the clichés–and their effects on the commitments stores of each group member.

Clichés may be seen as state transition machines that control the sequencing of dialog events between two participants in a conversation. A cliché restrains the unfolding of a conversation. Hence, it is possible to use specific clichés for reaching specific results in conversations with a well-known goal.

The structure for dialog representation called ACCORD [12] is used to represent the clichés in the context of this work. In this representation, a dialog is a sequence of events. Each participant has a commitment store containing the commitments undertaken by her during the conversation. The individual stores are accessible to other participants and are continuously updated while the conversation occurs.

According to the dialog primitives of ACCORD, a dialog event is a tuple consisting of a locution, and the speaker and the hearer of that locution. A locution is a locution modifier applied to one or more sentences. In the following subsections, the constructors of ACCORD conversation schema are presented.

Locution Modifiers and Dialog Events

Locutions consist of a statement and a modifier, represented as *locution modifier(statement)*. Statements are constructed in a propositional language, which includes negation, conditional, disjunction and conjunction of statements. Locution modifiers are as follows.

Assertions, to be read as "it is the case that s*tatement*", notationally ***asserts***(s*tatement*).

Questions, to be read as "is it the case that *statement*?", notationally ***questions***(*statement*).

Withdrawals, to be read as "I am not sure that *statement*", or "no commitment to s*tatement*", notationally ***withdraws***(s*tatement*).

Each locution modifier affects the commitment stores in a proper way, as will be seen later. There are four other locution modifiers that are not used in this paper [9].

Dialog events are represented by a notation in the form <*P1toP2, Locution*>, where *P1* and *P2* are participants. A dialog is formed by a sequence of dialog events, which is stored in an event register.

Commitments

Commitments are placed in commitment stores, which are represented in the following way: *Participant*(*C*(<*Set of sentences*>)*, D*(<*Set of sentences*>)). For the sake of clarity, each participant will not only have a commitment store for its positive and negative commitments–C part of the commitment store–but also a D part of the commitment store as a place for the statements which the participant is not committed to.

For example, $(P_1(C(s1,s2)) \wedge P_2(C(s1), D(s3))$ indicates that sentence *s1* is in part *C* of the commitment stores of participants P_1 and P_2, that sentence *s2* is in part *C* of P_1's commitment store and that sentence *s3* is in part *D* of P_2's commitment store. It is possible to read from that, that P_1 is committed to both *s1* and *s2*, while P_2 is committed to *s1* and is not committed to *s3*.

Commitments may be explicit or implicit. Explicit commitments are those generated in the dialog process, while implicit commitments are those inferred by the commitment calculus. Commitment stores are graphically represented according to Fig. 1.

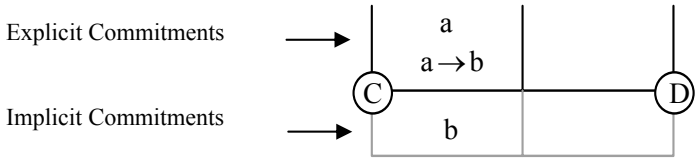

Fig. 1. Graphical representation of a commitment store (a and b are commitments)

Commitment Axioms

In this work, dialogs are represented in the following format:

$Pre \rightarrow [P_i toP_j, Locution]^n\ Post$

where *i* and $j \in \{1, 2\}, i \neq j$; $n \geq 1$, and *Pre* and *Post* are the conditions before and after the dialog. *Pre* and *Post* are commitments, and when they are instantiated as *healthy*, the current state of the dialog is valid.

Commitment axioms define the changes in the commitment stores caused by each locution modifier. Below commitment axioms are presented for some locution modifiers. P_1 and P_2 are participants and *s* is a <*statement*>.

$healthy \rightarrow [P_1 toP_2,$ ***asserts****(s)*$] P_1(C(s)) \wedge P_2(C(s))$

Given that *Pre* is a healthy state, after P_1 uttered asserts(s) to P_2, both participants are committed to s (this statement is included in part *C* of the commitment stores of both participants).

$P_1(C(s)) \rightarrow [P_1 to P_2,$ ***withdraws****(s)] $P_1(D(s))$*

Given that P_1 is committed to *s* prior to the assertion, she withdraws her commitment with such statement. This causes the removal of s from the *C* part of her commitment store and its inclusion in the *D* part.

healthy $\rightarrow [P_1 to P_2,$ ***questions****(s)] healthy*

Given that *Pre* is a healthy, after P_1 uttered *questions(s)* to P_2, *Post* is a healthy state. The uttering of a question does not affect the commitment stores.

In this work, clichés will be used as a means to realize the pre- and post-articulation phases. The coordination phase of articulation will use the model presented in the following.

2.2 Task/Interdependency Coordination Model

In order to realize the coordination process it is necessary to have a clear definition of tasks, collaborative activities and interdependencies. In the model adopted in this paper, a collaborative activity is a set of tasks carried out by several members of the group in order to achieve a common objective (commitments) [15]. Tasks are the building blocks of the collaborative activities and are connected by interdependencies. Tasks can be atomic or composed of sub-tasks. A group of sub-tasks can be considered a task on a higher abstraction level when it does not present interdependencies with tasks that are external to this group. This ensures the modeling of collaborative activities on several levels of abstraction.

Interdependency is a key concept in the coordination theory–if there are no dependencies between tasks to be performed in a collaborative effort, there is nothing to coordinate. (Note that this does not mean that there is no need for the pre- or post-articulation.) The approach task/interdependency is a step toward giving flexibility to coordination mechanisms, which is crucial to further use of this kind of mechanism. This approach is based on a clear separation between "the work devoted to activity coordination and coordinated work, i.e., the work devoted to their articulated execution in the target domain" [21].

One of the advantages of the separation task/interdependency is that interdependencies and their coordination mechanisms may be reused. It is possible to characterize different kinds of interdependencies and identify the coordination mechanisms to manage them, creating a set of interdependencies and respective coordination mechanisms capable of encompassing a wide range of collaborative applications [14]. An example of a set of coordination mechanisms that uses Petri Nets to model tasks and the treatment of interdependencies is found in [16].

The coordination can take place on two levels—the activities level (temporal) and the object level [7]. On the temporal level, the coordination defines the sequencing of the tasks that make up an activity. On the object level, the coordination describes how

to handle the sequential or simultaneous access of multiple participants through a same set of cooperation objects.

Temporal Interdependencies

Temporal interdependencies establish the relative order of execution between a pair of tasks. The set of temporal interdependencies of the proposed model is historically based on temporal relations defined by J. F. Allen [1]. He proved that there is a set of seven primitive and mutually exclusive relations that could be applied over time intervals. The adaptation of Allen's primitives to the context of collaborative activities takes into account that any task *T* will take some time (from *i* to *f*) to be performed.

A time interval is characterized by two events, which in turn are associated to time instants. The first event is the starting (initial) time of an interval *A*, denoted here *ia*. The other event is the ending (final) time of the same interval, denoted *fa*, always with *ia*<*fa*. Allen's relations between time intervals, *A* and *B*, are: **equals** (*ia*=*ib* and *fa*=*fb*), **starts** (*ia*=*ib* and *fa*<*fb*), **finishes** (*ia*>*ib* and *fa*=*fb*), **before** (*fa*<*ib*), **meets** (*fa*=*ib*), **overlaps** (*ia*<*ib*, *ib*<*fa* and *fa*<*fb*), **during** (*ia*>*ib* and *fa*<*fb*).

However, the merely descriptive characteristic of Allen's temporal relations allows for different interpretations of a single interdependency. For example, suppose that tasks *A* and *B* are associated by the **equals** relation. In this situation, what should the coordination mechanism do when task *A* is ready to begin, but not task *B*? Should it block the execution of task *A* until task *B* is ready (passive interpretation), or should it force the start of task *B* to guarantee that the interdependency will be respected (active interpretation)? In a different situation, if it is said that task *A* occurs before task *B*, what should be done when task *B* is ready but not task *A*? Should the coordination mechanism block task *B* until the end of task *A*, or should it allow the execution of task *B*, blocking future executions of task *A* (which would violate the relation)? For all of these reasons, it was necessary to make some adaptations to Allen's basic relations.

The first adaptation deals with active and passive interpretations by means of two operators: **enables** and **forces**. The **enables** operator represents the passive interpretation, while **forces** represents the active one. These operations may be applied on the initial and final instants of each interdependent task. Additionally, these extreme points have two states, *ready* and *concluded*, indicating, respectively, that the task is ready to start (or finish) and that it has already started (or finished). These states are used in the first operand, indicating that it will enable or force the second operand before (*ready*) or after (*concluded*) its own execution.

Consider, for example, two tasks *Ta* and *Tb*, with initial and final points *ia*, *ib*, *fa* and *fb*. The interdependency *Ta* **starts** *Tb*, which establishes that both tasks start simultaneously, may be extended into different interpretations:

ia (ready) **enables** *ib* AND *ib (ready)* **enables** *ia* – this statement indicates the passive situation, in which the tasks will start their execution only when both are ready (i.e., *Tb* will be enabled to start only when *Ta* is ready to start, and vice-versa), but neither will force the execution of the other.

ia (ready) **forces** *ib* – in this situation, when *Ta* is ready to begin, *Tb* is forced to start, indicating a master/slave active interdependency (similarly, *Tb* could be consid-

ered the master if *ib (ready)* **forces** *ia*).

In spite of the operators **enables** and **forces**, there are undefined situations remaining. Such a situation occurs, for example, in *Ta* **before** *Tb*. After *Ta* and *Tb* have been finished, how should the coordination mechanism proceed if *Tb* wants to start again? Should it allow its execution, since *Ta* has already been executed (one to many), or should it make *Tb* wait until *Ta* is executed again (one to one)?

In order to deal with such situations, it was necessary to include an optional parameter for the **enables** operator. This parameter indicates the number of times a condition (first operand) enables the event (second operand).

For example, to define that *Ta* **before** *Tb* allows the maximum of three executions of *Tb* for each execution of *Ta*, the following statements are used *fa (concluded)* **enables(3)** *ib*, indicating that, after each execution of *Ta*, *Tb* is allowed to execute up to three times. It is also possible to define that there is no restriction on the number of times a task may be executed after or during another.

In order to enhance the flexibility of the model, it is also necessary to create the **blocks** and **unblocks** operators that, respectively disable and re-enable the execution of an event (second operand) when the state of the first operand is reached. The use of these operators, for example, allows for a new interpretation of *Ta* **before** *Tb*:

ib (concluded) **blocks** *ia* – in this case, there is a restriction in the execution of *Ta*, which may not be executed anymore if *Tb* has already started its execution. There is no restriction on the execution of *Tb* (*Tb* does not have to wait for the execution of *Ta*, as would happen with the situation given by *fa (concluded)* **enables** *ib*).

Resource Interdependencies

Resource-related interdependencies may be represented by combinations of temporal relations. For example, if two tasks, *Ta* and *Tb*, may not use the same resource simultaneously, it is possible to define a "not parallel" dependency as the following statement, *ia (ready)* **blocks** *ib* AND *fa (concluded)* **unblocks** *ib* AND *ib (ready)* **blocks** *ia* AND *fb (concluded)* **unblocks** *ia*. However, besides being prone to deadlocks, this possibility ignores the notion of resource, which is quite important in the context of workflows and collaborative activities. Therefore, it is not sufficient to treat the problem of task interdependencies as a temporal logic problem. Considering resource management dependencies independently of temporal ones, a more flexible model is created, allowing the designer to deal with each kind of dependency separately.

Resource management interdependencies in the proposed model are complementary to temporal ones and may be used in parallel to them. This kind of interdependency deals with the distribution of resources among the tasks. Three basic resource management dependencies are defined.

Sharing – a limited number of resources must be shared among several tasks.

Simultaneity – a resource is available only if a certain number of tasks request it simultaneously. It represents, for instance, a machine that may only be used with more than one operator.

Volatility – indicates whether, after the use, the resource is available again. For example, a printer is a non-volatile resource, while a sheet of paper is volatile.

Each of the above interdependencies requires parameters indicating the number of resources to be shared, the number of tasks that must request a resource simultaneously and/or the number of times a resource may be used (volatility).

3 B-Web Example

B-webs are defined as "distinct systems of suppliers, distributors, commerce providers, infrastructure providers, and customers that use the Internet for their primary business communications and transactions" [23]. Different companies are increasingly reaggregating their values in these b-webs, which are becoming one of the driving forces of the digital economy.

A b-web is an adequate environment to apply the proposed articulation schema because it provides deeply interdependent tasks, which need computer-supported coordination. Moreover, the pre-articulation phase demands a normally bureaucratic conversation, which is suited to be represented by clichés.

The b-web typology comprises agoras, aggregations, alliances, value chains and distributive networks. Zooming into the value chain type of b-web, its main organization is called context provider, which "structure and direct the b-web network to produce a highly integrated value proposition" [23]. Other participants of this type of b-web may do everything else: manufacturing, delivering, on-site customer services, etc.

Value chains are further differentiated between routine production and shop production. While the former is product-centric and goods are designed for mass markets and production efficiency, the latter supports custom solutions where activities are not routine and are driven entirely by demand, that is, the end-customer is the one that triggers the value-creating process.

Cisco Systems is the quintessential example of shop production value chain. Using Cisco's Configuration Tool on the company's web page, the end-customer receives guidance to prepare its order while all kinds of discounts and other services are being offered. Only after selling the good does Cisco make it. However, in reality, Cisco will coordinate the production process instead of actually making it.

Cisco plays the coordinator role in this shop production value chain b-web. Using the available configuration tool customers communicate their orders, triggering the cooperation cycle among manufacturers, assemblers, distributors, component suppliers and the sales channels. The cooperation object itself is the computer (or solution) that will be shipped to the end-customer. Fig. 2 illustrates this process.

From the coordination point-of-view, it is possible to model the b-web workflow using the task/interdependency previously presented. An example of a simple transaction in a hypothetical value chain b-web is presented, involving four independent participants, a customer, the context provider, a producer, and a distributor. The workflow of this environment is represented in Fig. 3. In this figure, tasks are described in the ovals and their relations in the hexagons. Dotted arrows indicate interdependencies and normal arrows, workflow transitions. The word *or* in the workflows indicates alternative paths (only one of them is followed). The absence of *or* indicates parallel paths. Letters *a* and *b* near the inter-related tasks are used to identify

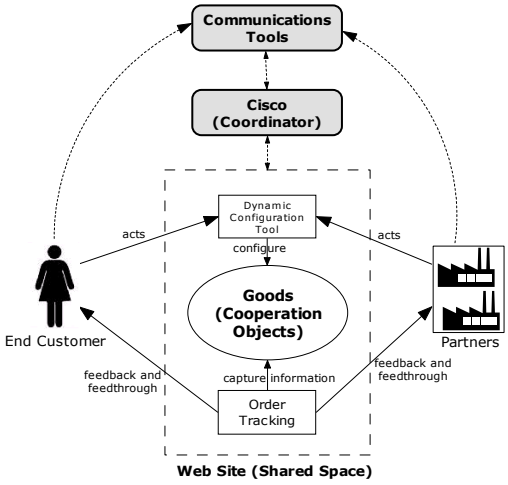

Fig. 2. Cisco Systems value chain b-web

them in the directives inside the hexagons.

In the workflow of Fig. 3, the customer must start the process by ordering the product in the context provider web site. This task starts the context provider's workflow (the end of the customer's task forces the beginning of the context provider's workflow). The customer then waits for the product or cancels the order. However, the order can only be cancelled if the product has not been delivered yet. This defines the relation between tasks deliver product (in the context provider workflow) and cancel order (in the customer workflow). The relation establishes that the beginning of the delivery blocks the order's cancellation.

After receiving an order, the context provider triggers two parallel activities: *confirm payment* and *order product*. This task starts the producer's workflow (*fa(concluded)* **forces** *ib*). After the payment confirmation and the ordering, the context provider must remain waiting until the product is ready. This determines the relation that when the producer concludes the product, task *organize delivery* in the context provider's workflow is enabled. We do not use the forces operator here because it is possible that the product becomes ready before the payment confirmation, and in this case, the delivery must wait, although enabled by the producer.

When the context provider is ready to organize the delivery, it starts the distributor's workflow, which has a close relationship to the rest of the context provider's one. This relationship is expressed by means of the *equals* relationship (*ia(ready)* **enables** *ib AND fa(ready)* **enables** *fb*), meaning that the tasks occur almost simultaneously in the distributor and in the context provider.

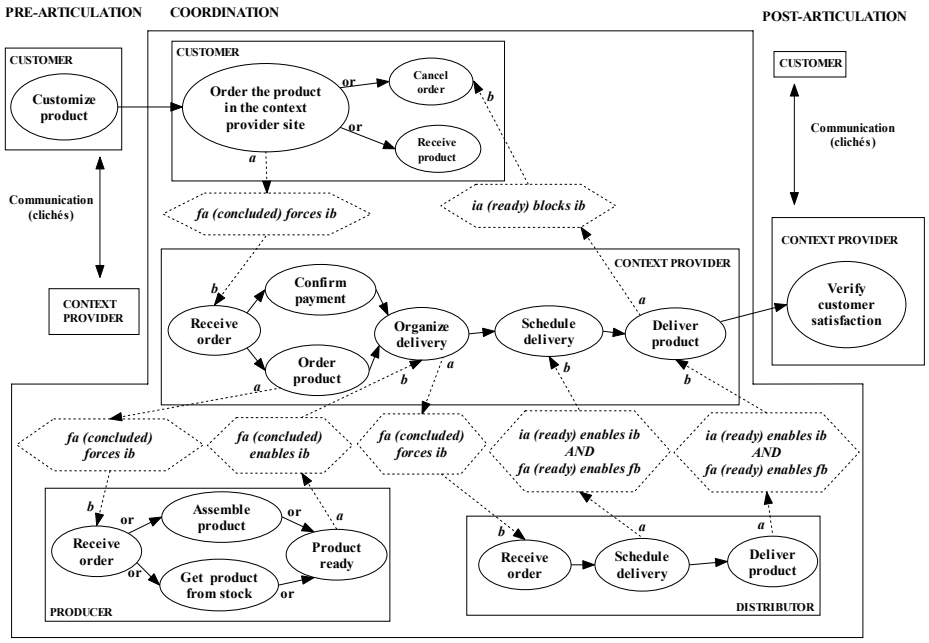

Fig. 3. Workflow of a typical transaction in a value chain b-web

The above example used only temporal interdependencies between tasks, but resource management dependencies could also be used in a straightforward way. For example, the context provider could have a stock, which defines an alternative route to that of contacting the producer. The task of getting the goods from the stock would have a volatility dependency, indicating that the stock would eventually finish.

The pre-articulation phase serves as an input to the workflow described above. The pre-articulation is part of the customer task *customize product*. During the performance of this task, the customer interacts with the company's web site and possibly to an operator. During this interaction, both the customer and the company assume commitments and, at end of the interaction, these commitments will define the product's configuration and delivery date. To illustrate this, consider the following situation.

The customer enters the web site and registers herself. To be registered, the customer accepts the terms and conditions of the company. After that, she enters the Dynamic Configuration Tool to order a product. She chooses product number AB123 and then, the Dynamic Configuration Tool offers configuration options. Then she chooses blue for the color and 220 for the voltage of the product. Then, the customer calls the site operator to negotiate the delivery date. She needs the equipment in three days. The operator says that it will be possible only for the green product. The customer accepts it, checks the price and orders the product.

The conversation cliché above is represented by the following sequence of dialog events and commitment stores.

I. Registration Dialog

healthy → [COMPANYtoCUSTOMER, **questions**(t)]
[CUSTOMERtoCOMPANY, **asserts**(t)]
CUSTOMER (C(t)) ∧ COMPANY (C(t))

II. Configuration Dialog

CUSTOMER (C(t)) ∧ COMPANY (C(t)) →
[CUSTOMERtoCOMPANY, **questions**(p)]
[COMPANYtoCUSTOMER, **asserts**(p)]
[CUSTOMERtoCOMPANY, **questions**(b)]
[COMPANYtoCUSTOMER, **asserts**(b)]
[CUSTOMERtoCOMPANY, **questions**(v)]
[COMPANYtoCUSTOMER, **asserts**(v)]
CUSTOMER (C(t,p,b,v)) ∧̣ COMPANY (C(t,p,b,v))

III. Delivery Date Negotiation Dialog

CUSTOMER (C(t,p,b,v)) ∧ COMPANY (C(t,p,b,v)) →
[CUSTOMERtoCOMPANY, **questions**(d)]
[COMPANYtoCUSTOMER, **withdraws**(b)]
[CUSTOMERtoCOMPANY, **withdraws**(b)]
[COMPANYtoCUSTOMER, **questions**(g)]
[CUSTOMERtoCOMPANY, **asserts**(g)]
[COMPANYtoCUSTOMER, **asserts**(d)]
CUSTOMER (C(t,p,v,g,d),D(b)) ∧̣ COMPANY(C(t,p,v,g,d),D(b))

IV. Ordering Dialog

CUSTOMER (C(t,p,v,g,d),D(b)) ∧ COMPANY (C(t,p,v,g,d),D(b)) →
[COMPANYtoCUSTOMER, **questions**(y)]
[CUSTOMERtoCOMPANY, **asserts**(y)]
CUSTOMER (C(t,p,v,g,d,y),D(b)) ∧ COMPANY (C(t,p,v,g,d,y),D(b))

Legend:
t – respect the company terms and conditions
p – manufacture the product AB123
b – manufacture the product in blue
v – manufacture the product for 220V
d – delivery the product in three days
g – manufacture the product in green
y – pay for the product AB123

At the end of the dialog, the customer is committed to respect the company terms and conditions and pay for the product. On the other hand, the company is committed to respect the terms and conditions, manufacture the product in 220V and in green color, and deliver it in three days for the stipulated price. These commitments serve as input to the workflow: pre-articulation. In a similar way, not shown here, it is possible to define a post-articulation cliché to verify customer satisfaction.

It is necessary to reinforce that this example deals with a single part of the whole b-web model. We did not stress all details for the modeled scenario (for instance, the consequences of the customer's cancellation, such as refund, devolution of the goods to the producer, and so on). Its main goal is to show how the representation models may be used in a practical situation.

One of the advantages of using the coordination model to represent the workflow of b-webs is that, by means of a formal mathematical analysis (for example, using Petri Nets [16], [24]), it is possible to anticipate and test the behavior of the interorganizational environments before their implementation. From the primitives of the coordination model, it is also possible to define a direct mapping to construct the adequate software coordination mechanisms [15].

4 The Articulation Schema

The articulation schema presented is illustrated in Fig. 4. In the pre-articulation phase, conversation clichés are used to generate user commitments regarding the tasks definition, tasks allocation, and other aspects necessary in the negotiation of the collaborative activity. These commitments serve to model the activity workflow, in which the interdependencies among tasks are coordinated. Finally, the execution of the activity leads to a post-articulation phase, in which the users may document and inspect the process, assess the quality of tasks execution, and realize other evaluations that generate inputs to following activities.

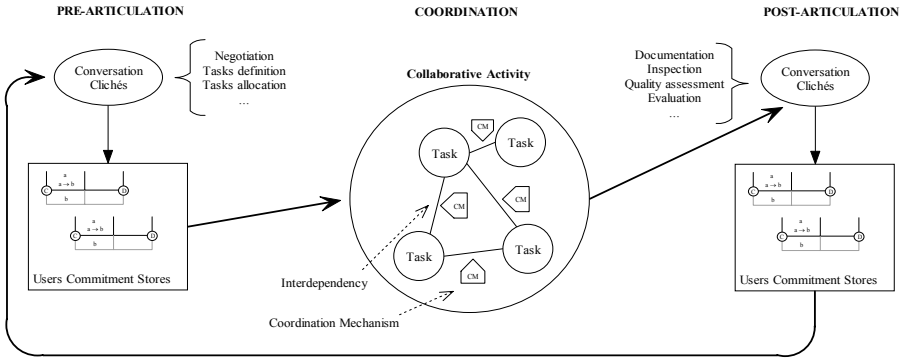

Fig. 4. The articulation schema

One of the important characteristics of this schema is the combination of features of different articulation approaches, namely those relying on communication–here represented by the conversation clichés– and those relying on explicit coordination mechanisms. The proposed articulation schema is discussed in the following sections in the light of some related works.

4.1 Articulation Dimensions

Carstensen and Nielsen, in a study comparing oral and artifact-based coordination, defined some dimensions that characterize articulation approaches [3]. In the light of such dimensions, we see that the presented articulation schema combines advantages of both coordination types:

Degree of automation: It is possible to implement the articulation schema presented here [12], [15].

Persistency: As an approach with high degree of automation, all the interactions may be stored.

Dedicated and non-exclusive coordination support: The pre- and post-articulation phases involve tasks that serve both the work and coordination, since it is more related to communication (non-exclusive coordination support). In the coordination phase, the work tasks are completely separated from coordination tasks (dedicated coordination support). Actually, this separation is the main characteristic of the task/interdependency model.

Direct and indirect referencing: The conversation clichés provide support for both direct referencing (i.e., telling someone what to do) and indirect referencing (i.e., providing information to assess what to do). This is also represented in the clichés by explicit and implicit commitments. The coordination model, on the other hand, provides only direct referencing, since it explicitly manages tasks interdependencies.

Dynamic and static: The pre- and post-articulation phases, once based on communication, tend to be dynamic, in the sense they may reflect changes in the world around the collaboration. The coordination model is essentially static.

Coupling and detachment: The pre- and post-articulation phases are detached from the field of work, in the sense that there is no straight connection between a state change in the coordination means and a state change in the work. The coordination mechanisms, on the other hand, are coupled to the field of work (tasks), in the sense that their state changes may affect the tasks (for example, when using the **forces** operator).

Reduction of coordination workload: The articulation schema augments the coordination workload during the pre- and post-articulation phases, since they use a structured conversation model. The task/interdependency model, on the other hand, reduces the coordination workload, because it provides the coordination mechanisms to regulate the tasks execution.

Flexibility: This coordination dimension needs further explanation. Therefore, it will be discussed separately, in the following section.

4.2 Articulation Flexibility

The necessity of coordination mechanisms to regulate interactions in collaborative systems has been the center of a heated discussion (e.g., [22], [26]). At one side, there are normative models that try to regulate the collaboration by restricting the interaction between participants and their tasks. A classical system that follows this approach is The Coordinator [25]. The criticisms on such normative approaches may be roughly summarized by the fact that their rigidly defined protocols applies only to very specific scenarios, limiting the flexibility of the collaborative systems. Eventually, there would be situations not predicted by the specified protocols, restraining the application of the defined mechanisms.

At the opposite site, are those advocating that collaborative systems should take flexibility to the extreme, leaving the articulation burden to the users, who become

extremely dependent on mutual awareness. The critics on this kind of approach are that they augment the coordination workload, since users must deal with the complexity of articulating their tasks. Moreover, giving the coordination responsibilities for the users does not guarantee that activities are performed according to any prescription.

Actually, the discussion makes sense because there are different kinds of collaborative activities. As previously mentioned, tightly integrated collaborative activities need coordination mechanisms to regulate interactions. On the other hand, the coordination workload and the limited flexibility imposed by such approaches are completely awkward for loosely integrated collaborative activities, which are more suited for the awareness-based approach.

In spite of this discussion, there is a trend to conciliate both ideas, arguing that both kinds of activities are "seamlessly meshed and blended in the course of real world" [20]. Moreover, it is argued that facilities should be provided so that the users may interpret and exploit predefined coordination standards, deciding to use, change or reject them [17].

Deiters et al., in a study about flexibility in workflow management systems, defined some flexibility dimensions and solutions proposed to achieve them [5]. Among their flexibility dimensions there are two that relates to the presented articulation schema:

Process flexibility: Related to situations when exceptions from the previously defined activities occur, and when there are parts of the activity that cannot be defined in advance. Regarding the presented articulation schema, the fact of having pre- and post-articulation phases based on communication attenuates the rigidity of the coordination phase, because exceptions and undefined situations may be renegotiated and reconsidered in a following coordination situation.

Flexible task allocation: Related to the reallocation of tasks at runtime. Deiters et al. [5] cited that one of the requirements for flexible task allocation is the negotiation of assignment rules and decisions. Moreover, the negotiation itself may be modeled as a workflow. The clichés-based pre-articulation phase presented here is an adequate means for such negotiation.

The articulation schema presented in this paper is definitely more related to tightly integrated collaborative activities, such as those that realize b-webs. However, the schema has some degree of flexibility, represented both by the separation between tasks and interdependencies in the coordination model and by the use of conversation during the pre- and post-articulation phases. Conversation, even in the form of clichés, is more flexible than coordination in the form of workflows.

5 Conclusion

This paper proposed an articulation schema based on two representation models, namely conversation clichés and task/interdependency model. Regarding the 3C model, the main contribution of this paper is to provide a step toward establishing an implemental linking between the structured aspects of communication and coordination. E-business, exemplified by b-webs, captures the structured side of the commu-

nication and coordination phases, being a proper application field for the articulation schema.

Considering communication, there is still room for the unstructured conversations that occur in the workplace. The same reasoning is applicable to coordination given that workflow only captures the structured side of coordination.

Although not explicitly discussed, the articulation schema does not disregard the role of awareness in articulation. Actually, awareness occupies a central position in the 3C model [10]. Every event taking place during communication, coordination and cooperation generates awareness information, which by its turn mediates all the aspects of the collaboration process. Awareness has a direct effect on the pre- and post-articulation phases, which are more dynamic, in the sense of reflecting the world's changes.

Acknowledgements

The authors are financed by individual grants awarded by the Brazilian National Research Council (CNPq): Hugo Fuks nº 303055/02-2, Alberto Raposo nº 305015/02-8 and Marco Gerosa nº 140103/02-3. Thanks to Prof. Carlos Lucena, Head of LES / PUC-Rio, and Prof. Marcelo Gattass, Head of Tecgraf / PUC-Rio, which is a group mainly funded by Petrobras, Brazilian Oil & Gas Company.

References

1. Allen, J.F., Towards a General Theory of Action and Time. Artificial Intelligence, 23, ACM, (1984) 123-154
2. Bond, A.H., A Computational Model for Organizations of Cooperating Intelligent Agents. Proceedings of Conference on Office Information Systems, SIGOIS Bulletin, 11(2- 3), ACM, (1990) 21-30
3. Carstensen, P.H., and Nielsen, M., Characterizing Modes of Coordination: A comparison between oral and artifact based coordination. Proceedings of GROUP, ACM, (2001) 81-90
4. Cohen, R.P., and Levesque, J.H., Persistence, Intention and Commitment. In Cohen, R.P., and Perrault, C.R. (eds.), Formal Theories of Communication, 171-203, Report nº CSLI-87-69, (1987)
5. Deiters, W., Goesmann, T., and Löffeler, T., Flexibility in workflow management dimensions and solutions. International Journal of Computer Systems Science & Engineering 15(5), CRL Publishing, (2000) 303-313
6. Ellis, C.A., Gibbs, S.J., and Rein, G.L., Groupware - Some Issues and Experiences. Comm. of ACM, 34(1), ACM, (1991) 38-58
7. Ellis, C.A., and Wainer, J., A Conceptual Model of Groupware. Proceedings of CSCW (1994), ACM Press, 79-88
8. Fuks, H., Ryan M., and Sadler, M., Outline of a Commitment Logic for Legal Reasoning. Proceedings of 3rd International Conference on Logics, Informatics and Law, V2, (1989) 391-405
9. Fuks, H., Negotiation using Commitment and Dialogue. PhD. Thesis, Department of Computing, Imperial College, University of London, (1991)

10. Fuks, H., Gerosa, M.A., and Lucena, C.J.P., The Development and Application of Distance Learning on the Internet. Open Learning - The Journal of Open and Distance Learning, 17(1), Carfax Publishing, (2002) 23-38
11. Koo, C., and Wiederhold, G., A commitment-based communication model for distributed office environments. Proceedings of Office Information Systems, SIGOIS Bulletin, 9(2-3), ACM, (1988) 291-298
12. Laufer, C.C., and Fuks. H., ACCORD: Conversation Clichés for Cooperation. Proceedings of COOP'95 – First International Workshop on the Design of Cooperative Systems, INRIA Press, (1995), 351-369
13. Malone, T.W., and Crowston, K., What is coordination theory and how can it help design cooperative work systems? Proceedings of CSCW, ACM, (1990) 357-370
14. Malone, T.W., and Crowston, K., The Interdisciplinary Study of Coordination. ACM Computing Surveys, 26 (1), ACM, (1994) 87-119
15. Raposo, A.B., and Fuks, H., Defining Task Interdependencies and Coordination Mechanisms for Collaborative Systems. In Blay- Fornarino, M. et al. (eds.), Cooperative Systems Design. Frontiers in Artificial Intelligence and Applications, 74, IOS Press, (2002) 88-103
16. Raposo, A.B., Magalhães, L.P., and Ricarte, I.L.M., Petri Nets Based Coordination Mechanisms for Multi-Workflow Environments. International Journal of Computer Systems Science & Engineering 15(5), CRL Publishing, (2000) 315-326
17. Schmidt, K., Riding A Tiger, Or Computer Supported Cooperative Work. Proceedings of The 2nd European Conference on Computer-Supported Cooperative Work, Kluwer, (1991) 1-16
18. Schmidt, K., and Bannon, L.J., Taking CSCW seriously - Supporting articulation work. Computer Supported Cooperative Work, 1(2), Kluwer, (1992) 7-40
19. Schmidt, K., and Simone, C., Coordination mechanisms: Towards a conceptual foundation of CSCW systems design. Computer Supported Cooperative Work, 5(2-3), Kluwer, (1996) 155-200
20. Schmidt, K., and Simone, C., Mind the gap! Towards a unified view of CSCW. Proceedings of COOP 2000 - 4th International Conference on the Design of Cooperative Systems, IOS Press, (2000) 205-221
21. Simone, C., Mark, G., and Giubbilei, D., Interoperability as a Means of Articulation Work. Proceedings of WACC: Int. Conf. on Work Activities Coordination and Collaboration, ACM, (1999) 39-48
22. Suchman, L.A., Do Categories Have Politics? Computer Supported Cooperative Work, 2(3), Kluwer, (1994) 177-190
23. Tapscoot, D., Ticoll, D., and Lowy, A., Digital Capital: Harnessing the Power of Business Webs. Harvard Business School Press, USA, (2000)
24. van der Aalst, W.M.P., The Application of Petri Nets to Workflow Management. The Journal of Circuits, Systems and Computers, 8(1), World Scientific, (1998) 21-66
25. Winograd, T., and Flores, F., Understanding Computers And Cognition. Addison-Wesley, USA, (1987)
26. Winograd, T., Categories, Disciplines, and Social Coordination. Computer Supported Cooperative Work, 2(3), Kluwer, (1994) 191-197
27. Woo, C.C., Sact - A Tool for Automating Semi-Structured Organizational Communication. Proc. of Conference on Office Information Systems, ACM, (1990) 89-98