The 7th International Conference on Computer Supported Cooperative Work in Design - CSCWD 2002, p.25 - 30. Rio de Janeiro, Brazil. September 25-27, 2002. W. Shen, J. M. de Souza, Z. Lin, J.P. Barthès (Eds.). ISBN: 85-285-0050-0.

Coordination in Collaborative Environments – A Global Approach

Adailton J. A. da Cruz
[1]*DCA – FEEC*
Univ. of Campinas - Brazil
[2]*CEUD – UFMS - Brazil*
ajcruz@dca.fee.unicamp.br

Alberto B. Raposo
Tecgraf
Computer Science Dept.
PUC-Rio - Brazil
abraposo@tecgraf.puc-rio.br

Léo P. Magalhães
DCA – FEEC
Univ. of Campinas - Brazil
leopini@dca.fee.unicamp.br

Abstract

In this work we present a methodology to express both analytically and graphically the interdependencies among tasks realized in a collaborative environment. For each interdependency expression, a coordination mechanism is built, modeling the global behavior of the environment, i.e., the structure that ensures the realization of the tasks according to the established interdependencies.

1. Introduction

An important aspect of collaborative work is the notion of tasks interdependencies [5]. These interdependencies are normally positive, in the sense that each participant wants the works of others to succeed. However, they are not always harmonious. It is necessary for coordination between tasks to exist in order to guarantee the efficiency of the collaboration. Without coordination, there is the risk that participants would get involved in conflicting or repetitive tasks. Coordination, in this context, is defined as "the act of managing interdependencies between activities performed to achieve a goal" [3].

In this sense, coordination in collaborative environments is managed by coordination mechanisms, defined as a "coordinative protocol with an accompanying artifact, such as, for instance, a standard operating procedure supported by a certain form" [7].

One of the main challenges related to the coordination of collaborative activities is to develop coordination mechanisms that are, at the same time, global and localized. The global part of the coordination mechanism should be able to collect, from the complex relationships among tasks in the collaborative environment, all the conditions that enable or not the beginning of the tasks. The localized part should be responsible to coordinate the pre-authorized execution of tasks, following the kind of relationship established for these tasks.

This paper introduces a methodology called RG (Relationships Graph) that describes, analytically and graphically, the relationships among collaborative tasks and constitutes the basis upon which global coordination mechanisms can be assembled from localized ones. The next section discusses some general issues regarding coordination mechanisms. The RG methodology is presented in Section 3 and a set of coordination mechanisms built using this methodology is presented in Section 4. Section 5 presents the conclusions of this work.

2. Temporal Coordination Mechanisms

The nature of the coordination (i.e., temporal, causal, etc.) is established by the set of relationships allowed among tasks. For example, if those relationships are temporal, then the tasks' behaviors are coordinated in relation to their execution times (e.g., task A must be executed *before* task B and *at the same time of* task C). On the other hand, if the relationships are causal, then tasks are coordinated based on events (e.g., if task A occurs, then task B and task C must also occur).

The execution of an activity occurs in a time interval and may be considered as the result of a set of tasks (atomic actions). Consequently, these tasks are related by the time parameter. The non-execution of one or more tasks may harm the execution of the whole activity, depending on the degree of interdependency of the involved tasks.

The coordination of the realization of interdependent tasks in collaborative environments requires mechanisms that should be able to model those interdependencies and ensure that they will not be violated, enabling that the collaborative activity be fully, partially, or not executed. If all the tasks that compose the activity are authorized to execute, then the activity is fully executed. The coordination mechanism authorizes a task only when its realization does not violate any defined interdependency.

In the following sections we are going to present a methodology to describe activities and to procedurally obtain the coordination mechanisms from these descriptions.

3. The RG Methodology

The RG methodology consists in obtaining a set E of expressions that describes, from a set of relations (primitives) R, the interdependencies in a group of tasks. These tasks are related to each other composing a collaborative activity. This approach does not restrict the number of relations a task may have with another.

Examples of activities adequate for this methodology are the construction of a multimedia presentation, and the assembly of a product composed of several pieces that become available during the assembly process. In the first case, tasks may be the presentation of texts, the reproduction of audios, videos, images, among others, which need to be synchronized. In the second example, tasks are the connections of the different pieces.

Once the relations among tasks are described by the set E, it is necessary to define the coordination mechanism for each expression $e_i \in E$. The elements of E have both an analytical and a graphical representation. In the following sections it will be shown how an expression e_i may be mapped onto a connex graph.

3.1. The Set of Relationships in RG

The set of relations R adopted in this work is based on the temporal logic proposed by Allen [1]. He proved that there is a set of primitive and mutually exclusive relations that could be applied over time intervals (i.e., any pair of time intervals are necessarily related by one and only one of Allen's relations). From Allen's first order predicate logic, we have chosen the 7 primitives that constitutes R (Fig. 1).

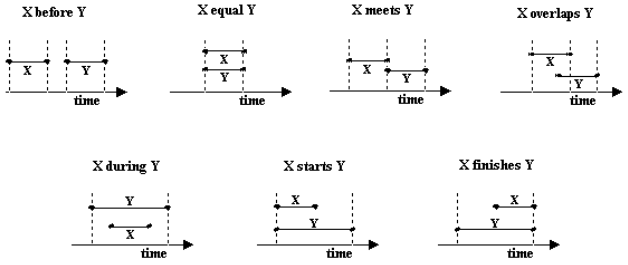


Figure 1. Set *R* of the 7 relations presented in [1].

The fact of being applied over time intervals (and not over time instants) made the relations of Fig. 1 suited for task coordination purposes, because tasks, although being the atomic actions of collaborative activities, are non-instantaneous operations.

3.2. The Expressions in RG

The primitives of R are binary relations, i.e., they relate only two tasks. However, it is possible to overcome this limitation, allowing that a single task be related to k other tasks by means of those primitives.

In order to describe an expression formed by n tasks t_1, t_2, …, t_n in the RG methodology we adopt the notation of the Graph Theory used by [8] as described below.

Definition 1: An expression *e(T,R)* is composed of a finite non-empty set T of tasks and a set R of labeled ordered pairs with distinct elements of T. The labels of the elements of R are defined by associating them to one of the following unitary sets, which respectively indicates the relations *before*, *during*, *equal*, *finishes*, *meets*, *overlaps* and *starts*: *{b}*, *{d}*, *{e}*, *{f}*, *{m}*, *{o}* and *{s}*.

The expression *e(T,R)* has a graphical representation where tasks $t_i \in T$, $i = 1, 2, \ldots, n$, correspond to distinct points of the plane located in arbitrary positions. The elements of R correspond to labeled arcs that connect two distinct points of the plane given by the ordered pair. Fig. 2 illustrates an example for n=12.

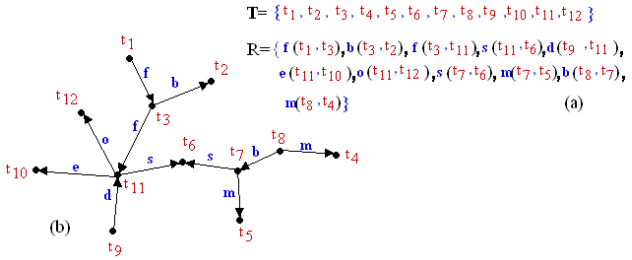


Figure 2. (a) An expression *e(T,R)*. (b) Its graphical representation

In an expression *e(T,R)* a task t_i has a degree k if it is related with k other tasks. For instance, in Fig. 2, *degree*(t_3) = 3.

The expression *e(T,R)* is called cyclic if there exists a sequence of k tasks, $1 \leq k < n$, where we start in a task t_i and return to it following that sequence in the graph. In this paper we are investigating acyclic expressions.

An expression *e(T,R)* does not generates an inconsistency if it is acyclic, i.e., the n-1 primitives do not generate an unrealizable configuration. This fact is guaranteed by the mutual exclusion and exhaustiveness properties of the primitives of R [9]:

Property 1: Given two tasks, X and Y, there is an $r_i \in R$ such that $X\ r_i\ Y$ or $Y\ r_i\ X$ is true (exhaustiveness).

Property 2: If $X\ r_i\ Y$, where $r_i \in R$, then there is no $r_j \neq r_i$, $r_j \in R$, such that $X\ r_j\ Y$ is true (mutual exclusion).

4. Coordination Mechanisms for Expressions

The coordination mechanism for an expression in E is constructed in two phases. In the first one, called Global Coordination Mechanism (GCM), the conditions that must be satisfied to authorize the beginning of the tasks are modeled. These conditions are established by the

primitives that define the expression. In the second phase, called Local Control Mechanism (LCM), it is modeled the mechanism that guarantees the execution of the temporal relation established for two tasks.

Analyzing the example in Fig. 2, we can see that, for example, task t_{12} is related only to t_{11} (degree(t_{12})=1) and, therefore, does not need to satisfy any global condition. However, degree(t_{11})=5, which means that this task must satisfy some global conditions. For example, the beginning of t_{11} must be authorized simultaneously to the beginning of t_{10} (since t_{11} equal t_{10}), t_6 (t_{11} starts t_6) and t_7 (t_7 starts t_6). Another global condition for t_{11} is that it must start some time after the end of t_8 (since t_8 before t_7, and the beginning of t_7 must be simultaneous to that of t_{11}, as shown above). This is enough to give an idea that the determination of global conditions may not be easily accomplished only by looking at the graph of *e(T,R)*. The following sections will present a procedural approach to find the global conditions.

4.1. GCM – Global Coordination Mechanism

The process of a GCM construction adopts an "outside-in" approach, i.e., it starts with the most external tasks (degree=1), going to the most internal ones (higher degrees). This approach is justified because tasks with degree=1 do not need to satisfy global conditions and generally represent the majority of tasks in an expression.

Hence, given an expression e_1, we obtain a new expression e_2 by eliminating all tasks with degree=1. The derived expression e_2 also has a set of tasks with degree=1. Geometrically, we have a star of *k* legs, where the center is a task with degree=k and the legs are the arcs connecting it with its *k* related tasks. A star is defined as a sub-expression of *E* corresponding to a task *t* related to *k* other tasks (*degree(t)=k*) – see Fig. 3.

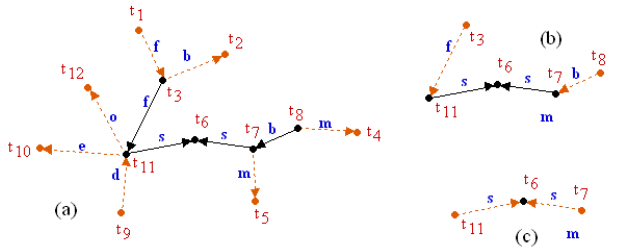


Figure 3. (a) First iteration, indicating tasks with degree=1 that will be removed from the expression. (b) and (c) following iterations.

The next step is to determine the global conditions for the tasks of degree=1 in e_2 regarding the *k* tasks related to them. In the example of Fig. 3, these tasks are t_3 and t_8.

The next iteration repeats the same steps, i.e., from e_2 we obtain an expression e_3 eliminating all tasks with degree=1 in e_2. Then we determine the global conditions for the tasks with degree=1 in e_3 regarding the *k'* tasks related to them, excepting those representing the center of the stars analyzed in the previous iteration. For example, in Fig. 3 (c), which corresponds to the expression e_3, the tasks with degree=1 are t_{11} and t_7. From the *k'* tasks related to t_{11}, task t_3 is the center of the star analyzed in the previous iteration.

In the second iteration two levels of stars appear; a more external one, generated in the first iteration and a more internal star, generated in the second iteration. In Fig. 3, the first iteration generates stars with centers t_3 and t_8, while the second iteration generates stars with centers t_{11} and t_7. Continuing with the process, we need to connect the stars of these both levels, by considering the relation with the centers of two adjacent stars.

The iterations are executed until the expression e_i, $i<n$, has one or two tasks. It can be demonstrated that this process is consistent and always finishes with one or two tasks. The algorithm for this process is described below:

Given an expression e(T,R) with n tasks;
Determine the degree of all tasks;
While there is a task t with degree(t)=1
  Eliminate tasks t with degree(t)=1;
  Generate stars for tasks t' with degree(t')=1;
  Connect these stars with stars of the previous iterations;

4.1.1. Global Conditions. Global conditions (GCs) are conditions imposed to a task in order to guarantee the logic of the primitives that compose the expression. For example, the beginning of t_{11} in the expression of Fig. 3 (a) must be authorized simultaneously with the beginning of t_6, t_7 and t_{10}.

The determination of GCs follows the algorithm presented in the previous section, i.e., it is established the global conditions for the stars generated in iterations *i* and *i+1*, and then the conditions corresponding to the connections of adjacent stars.

The adopted strategy for the determination of GCs is to construct a map *M* with all possible global conditions. This way, based on *M*, it is possible to determine the conditions that are pertinent to any relation in a star.

The map *M* is elaborated evaluating 14 forms of relationships possible for a task *a*. These possibilities corresponds to the 7 primitives of *R* and their respective inverse relationships. This occurs because $r(a,b) \neq r(b,a)$. For example, *a before b* is different from *b before a*. The only exception is when *r=equal*. Fig. 4 describes these forms of relationships, where the order in which the tasks are related are indicated by the arrow.

The GCs related to task *a* are more easily identified if the tasks related to it are positioned, according to their respective interdependencies, over a timeline. This process generates a consistency graph, as shown in Fig. 5.

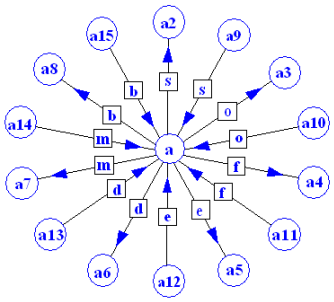


Figure 4. All possible relations involving a task.

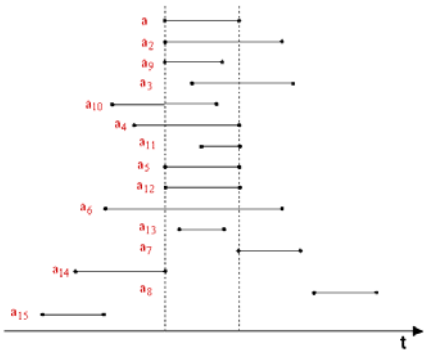


Figure 5. Consistency graph for a task.

Based on the consistency graph for task *a*, as shown in Fig. 5, the list of GCs used in the specification of the map M is described below:

c1) *a* must start simultaneously to *a2*;
c2) *a* must start simultaneously to *a9*;
c3) *a3* may start if *a* has already started;
c4) *a* may start if *a10* has already started;
c5) *a* may start if *a4* has already started;
c6) *a11* may start if *a* has already started;
c7) *a* must start simultaneously to *a5*;
c8) *a12* must start simultaneously to *a*;
c9) *a* may start if *a6* has already started;
c10) *a13* may start if *a* has already started;
c11) *a* may start if *a7* is ready to begin, but *a7* may only start at the end of *a*;
c12) *a14* may start if *a* is ready to begin, but *a* may only start at the end of *a14*;
c13) *a8* may start only after the end of *a*;
c14) *a* may start only after the end of *a15*;
c15) the end of *a* must be simultaneous to that of *a4*, *a5*, *a11*, and *a12*;
c16) the conditions to the end of the other tasks are controlled by their LCMs.

4.2. LCM – Local Coordination Mechanism

The LCM connects two related tasks, guaranteeing that a single interdependency between them will not be violated. The LCM is responsible for telling each task when it may or must start and finish its execution. It is called "local" because it coordinates only one relation between two tasks, without knowledge of GCs related to each of them. In order to implement the LCMs we use a set of Petri net-based coordination mechanisms proposed in [4].

A detailed explanation of the LCMs is out of the scope of this paper. What is important here is that LCMs may be viewed as black boxes connecting two tasks in map *M*. In the Petri net-based map, each of the possible forms of relationship of a task may be modeled according to the schema presented in Fig. 6.

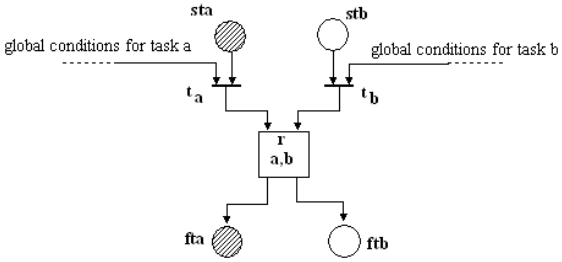


Figure 6. Schema to connect two related tasks.

In the schema of Fig. 6, task *a* is the center of the star and has a temporal relation *r(a,b)* or *r(b,a)* with task *b*. In this schema, task *b* is associated to two places, one transition and a box representing the LCM for this relation. Place *stb* indicates that *b* is ready to start and place *ftb* indicates the end of its execution. Transition *tb* receives all the conditions that *b* must satisfy and, when fired, indicates to the LCM that the global conditions for *b* are satisfied and it is authorized to begin. The LCM then assumes the coordination of the relation between *b* and the center of the star (task *a*).

4.3. The Coordination Map

The map *M* is the Petri net representation of all possible relationships involving a task *a* (as in Fig. 4). Its main goal is to show the "worst case" situation, from which other situations may derived. The map (Fig. 7) is constructed by applying the schema of Fig. 6 and the list of GCs (Section 4.1.1) to each of the 14 tasks related to task *a*.

In *M*, places sa_i indicate that task a_i is ready to start, and places fa_i indicate the end of a_i. The firing of transition tsa_i authorizes the beginning of a_i.

Transition t_{1sa} models conditions c1, c2, c7, and c8 (Section 4.1.1). The arc $(sa7,t_{1sa})$ indicates that *a7* is ready to start (part of condition c11). The arc $(t_{1sa}, Pa14)$ ensures that *a14* will start only when *a* is ready to start (part of c12). Transitions t_{sa10}, t_{sa4} and t_{sa6} respectively indicate to *a* that *a10*, *a4* and *a6* have already started (conditions c4, c5 and c9).

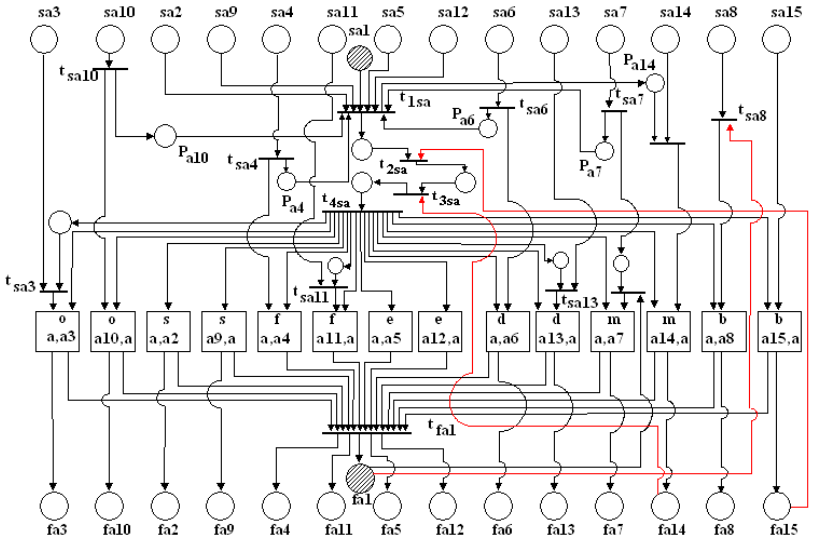


Figure 7. The map *M*.

Transition t_{2sa} implements condition c14, and transition t_{3sa} implements the second part of c12. The firing of t_{2sa} must occur before the firing of t_{3sa} because task *a15* finishes before *a14*, as may be visualized in the consistency graph (Fig. 5). Tansition t_{4sa} authorizes simultaneously the beginning of task *a* at all LCM involved with it.

Transitions t_{sa3}, t_{sa11} and t_{sa13} respectively control the global conditions necessary for the beginning of *a3*, *a11*, and *a13* (conditions c3, c6 and c10, respectively).

Transition t_{sa7} authorizes the beginning of *a7* as soon as *a* finishes. (second part of condition c11). Transition t_{sa8} authorizes the beginning of *a8* if *a* has already finished (condition c13). Finally, transition t_{fa1} synchronizes the end of *a* with the end of *a4*, *a5*, *a11*, and *a12*.

The map *M* considers one instance of each possible relation to task *a*. However, two situations may occur: i) *a* does not have all of the 14 relations, ii) *a* is related with different tasks, but with the same kind of relation. In the first situation, the GCM is obtained by reproducing map *M* only with the existent relations. In the second situation, it is necessary to repeat the implementation of the conditions pertinent to the primitive that appears more than once. Fig. 8 (a) and (b) shows the GCMs of the stars *t3* and *t11* of the expression presented in Fig. 2.

Finally, it is necessary to consider the connection two adjacent stars. Suppose that a star *a* is generated in iteration *i* and a star *b* is generated in iteration *i+1*. Therefore, the relation connecting both stars (i.e., *r(a,b)* or *r(b,a)*) is modeled in the GCM of *a* and, following the schema of Fig. 6, there is a transition t_b that models the GC of *b* in relation to the tasks of star *a*. Similarly, there is in the GCM of *b* another transition t_b that models the GCs of *b* in relation to the tasks of star *b*. These transitions t_b, when fired, authorize the execution of *b*. Hence, to connect both stars, it is necessary to merge transitions t_b. This is done by eliminating t_b from the GCM of *a* and redirecting the GCs to t_b of the GCM of *b*, creating a single t_b that authorizes the LCM to execute *b*. Fig. 8 (c) shows the GCM resulting from the connection of stars *t3* and *t11*, and (d) shows the whole GCM for the expression of Fig. 2.

5. Conclusion

We present a methodology to describe and coordinate interdependencies of sets of tasks in collaborative environments. Based on the description, the algorithm to obtain the coordination mechanisms is shown.

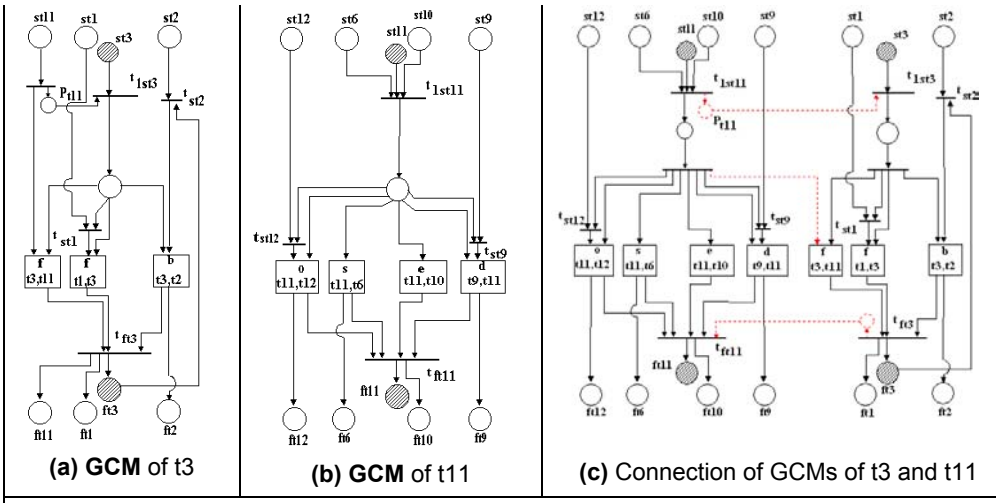
Among the contributions of this methodology, we can cite the global treatment given to the coordination process, i.e., a task is executed only if no relation of interdependency is violated.

Another approach to group coordination based on partial knowledge of the tasks structure of the environment is presented in [2]. Its goal is to create a framework for the coordination of a group (involving human and software agents) to guarantee that the tasks are completely solved in a timely and efficient manner.

The use of concepts from the graph theory allowed an effective approach in the modeling of the global conditions, generating a standard procedure that is independent of the number of tasks taking part in the collaborative activity. This is a further step in previous approaches that limit the number of tasks to which a task may be related (e.g., [5]).

The consistency graph used for the identification of the global conditions may also be used to verify if a set of tasks is realizable, i.e., if it does not generate an inconsistent configuration.

The relations graph offers a map of the whole



(a) GCM of t3

(b) GCM of t11

(c) Connection of GCMs of t3 and t11

(d) Compete GCM for the expression of Figure 2.

Figure 8. The construction of a GCM.

collaborative activity and this, in conjunction with techniques from the graph theory, may be used for analysis processes in order to decide for example what will happen if a certain task is not executed.

Finally, it is important to reinforce that not all cycles in the relations graph generate inconsistencies. One of the next steps of this work is to investigate properties that could enable the existence of cycles in the graph. Another future work is to create software components that implement the GCMs, based on the LCM implementation presented in [6].

6. References

[1] J. F. Allen, "Towards a General Theory of Action and Time", *Artificial Intelligence*, 23, 1984, pp. 123-154.

[2] K. S. Decker, "Coordinating Human and Computer Agents", In W. Conen, and G. Neumann (eds.), *Coordination Technology for Collaborative Applications – Organizations, Processes, and Agents*, LNCS 1364, Springer-Verlag, 1998, pp. 77-98.

[3] T. W. Malone, and K. Crowston, "What is Coordination Theory and How Can It Help Design Cooperative Work Systems?", *Conf. on Computer-Supported Cooperative Work (CSCW'90)*, 1990, pp. 357-370.

[4] A. B. Raposo, L. P. Magalhães, and I. L. M. Ricarte, "Petri Nets Based Coordination Mechanisms for Multi-Workflow Environments", *Int. J. Computer Systems Science & Engineering*, 15(5), .2000, pp. 315-326.

[5] A. B. Raposo, L. P. Magalhães, I. L. M. Ricarte, and H. Fuks, "Coordination of Collaborative Activities: A Framework for the Definition of Tasks Interdependencies", *7th Intl. Workshop on Groupware (CRIWG'2001)*, 2001, pp. 170-179.

[6] A. B. Raposo, A. J. Cruz, C. Adriano, and L. P. Magalhães, "Coordination Components for Collaborative Virtual Environments", *Computers & Graphics*, 25(6), 2001, pp. 1025-1039.

[7] C. Simone, and K. Schmidt, "Taking the distributed nature of cooperative work seriously", *6th Euromicro Workshop on Parallel and Distributed Processing*, 1998, pp. 295-301.

[8] J. L. Szwarcfiter, *Grafos e algoritmos computacionais*, Editora Campus, Rio de Janeiro, 1986.

[9] A. K. Zaidi, "On Temporal Logic Programming Using Petri Nets", *IEEE Trans. Systems, Man, and Cybernetics – Part A: Systems and Humans*, 29(3), 1999, pp. 245-254.