In: Computers and Advanced Technology in Education (CATE'2002), pp. 20-25. G. A. S. Torrellas and V. Uskov (Eds.), Acta Press, 2002. ISBN 0-88986-332-6.

TASKS INTERDEPENDENCIES IN COLLABORATIVE LEARNING ACTIVITIES: SPECIFICATION AND MODELING

Ivan L. M. Ricarte
State University of Campinas (UNICAMP)
School of Electrical and Computer Engineering (FEEC)
Dept. of Computer Eng. and Industrial Automation (DCA)
CP 6101 – 13083-970 – Campinas, SP, Brazil
ricarte@dca.fee.unicamp.br

Alberto B. Raposo
Computer Graphics Group (Tecgraf)
Catholic University of Rio de Janeiro (PUC-Rio)
R. Marquês de S. Vicente, 225
22453-900 – Rio de Janeiro, RJ, Brazil
abraposo@tecgraf.puc-rio.br

ABSTRACT

In collaborative environments, coordination is an essential matter to the specification of activities, which are described as sets of interdependent tasks. An extensible framework encompassing coordination mechanisms to specify tasks interdependencies is initially presented. These mechanisms are formally modeled using Petri nets and, along with a Petri net representation for the tasks to be performed, can be used to create a model to evaluate the environment. The use of this framework is illustrated in the context of a learning environment based on collaborative Web document construction.

KEY WORDS

Collaborative Knowledge Construction and Learning, Computer Supported Collaborative Learning, Coordination, Petri Nets.

1. INTRODUCTION

Planning is an essential activity to collaboratively develop 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 cooperative work" [1]. Articulation includes activities such as identification of objectives, establishment of tasks mapped from the objectives, and selection of participants to perform the tasks. Result of such activities, once established, rarely is changed. Coordination is a part of the articulation work with a distinct, dynamic nature.

Coordination has been defined as "the act of managing interdependencies between activities performed to achieve a goal" [2]. Thus, it is essential to assure that the performed tasks are indeed part of a larger collaborative effort, avoiding execution of conflicting or repetitive tasks. Due to its dynamic aspect, it can be considered the most demanding activity of the articulation work.

Coordination theory was originally related to collaborative work. Many computer-based environments support collaboration by providing access to computer-mediated communication tools, expecting or externally enforcing that such tools are used to build the learning community. However, not always communication tools are effective by themselves, and coordination mechanisms can help to structure and organize the performance of subtasks.

In this paper, a framework to define generic interdependencies that occur between tasks in collaborative activities is presented. Petri nets are introduced and used to formally present the constructs in this framework. The use of the framework is illustrated by the description of the coordination model for the realization of a collaborative activity, composing a Web document, in a learning environment.

2. A FRAMEWORK TO DEFINE TASKS INTERDEPENDENCIES

Interdependency is a key concept in coordination theory: If there is no dependencies between tasks performed in a collaborative effort, there is nothing to coordinate [3]. A particular view of coordination is related to workflow management, in which interdependencies are related to the occurrence and temporal order of events. Although usually related to the general notion of a business process (see e.g. [4]), the concept of workflow has also been applied to educational environments [5].

In this work, a collaborative activity is a coordinated set of tasks realized by multiple actors in order to achieve a common goal. A task, either atomic or a group of subtasks without external dependencies, is a "building block" of collaborative activities. This definition enables the modeling of collaborative activities using several abstraction levels for coordination specification and

management. Interdependencies in the framework are either temporal, when related to the tasks execution order, or resource management, when related to resource distribution among tasks [6].

2.1. TEMPORAL INTERDEPENDENCIES

Temporal interdependencies establish the relative order of execution between a pair of tasks, considering precedence or concurrency relationships. The set of temporal interdependencies was inspired by a classical work of J. F. Aleen on temporal relations over time intervals [7].

A time interval is characterized by two events, associated with time instants. The first event is the starting time of an interval *x,* denoted here ti_x. The other event is the ending time of the same interval, tf_x, always with $ti_x < tf_x$. According to Allen, a set of seven primitive and mutually exclusive relations maintains temporal information considering any pair of time intervals *X* and *Y*:

X equals Y when $ti_x = ti_y$ *and* $tf_x = tf_y$;

X meets Y when $tf_x = ti_y$;

X starts Y when $ti_x = ti_y$ *and* $tf_x < tf_y$;

X finishes Y when $ti_x > ti_y$ *and* $tf_x = tf_y$;

X before Y when $tf_x < ti_y$;

X during Y when $ti_x > ti_y$ *and* $tf_x < tf_y$.

X overlaps Y when $ti_x < ti_y$ *and* $ti_y < tf_x$ *and* $tf_x < tf_y$;

Adaptation of these primitives to the context of collaborative activities takes into account that any task T will take time from ti to tf to be performed. However, in specifying temporal interdependencies between two tasks not always one should be concerned with both start and finish events, and some relaxed relationships are introduced, thus comprising thirteen coordination mechanisms between two tasks T1 and T2.

T1 equals T2 establishes that two tasks must be executed simultaneously, which can be interpreted in two ways. The active interpretation of this coordination mechanism expresses that the beginning of T1 fires T2; similarly, finishing T1 concludes T2. Consider a situation in which T1 denotes a discussion session and T2 its “recording”. From the coordination point of view, this “active-equals” relationship between these tasks would simply indicate that the T2 should follow the execution of T1. However, a problem to proceed with session recording would not invalidate the session itself, which could proceed to its conclusion. The passive interpretation expresses a set of conditions that should be obeyed to proceed with tasks execution. In the same example, this would be the case in which the session recording must be ready before the start of the discussion. A problem in recording would delay the beginning of the discussion session until the problem is solved.

T1 meets T2 expresses that T2 should start immediately after the end of T1. This relation could be used to restrict a question-and-answering session (T2) to occur only and immediately after an exposition session (T1).

T1 startsA T2 and *T1 startsB T2* express that two tasks should start at the same time. *T1 startsA T2* preserves Allen's semantic, i.e., T1 and T2 start at the same time but T1 should finish before T2. In a laboratory activity, T1 might represent an experiment that should be concluded before its report (T2). The dependency *T1 startsB T2* is a relaxed coordination construct, useful to model that two tasks should start at the same time but the relationship between their finishing times is not relevant. For example, a group coordinator may want to assign tasks with the same starting time to group members without regard to their finishing order.

Dependencies *finishes* express that tasks should conclude at the same time. *T1 finishesA T2*, as Allen's relation, indicates that T1 starts after T2 and both should finish together. *T1 finishesB T2* relaxes any restriction regarding starting times. This would be the case when T2 represents the controller task of an online lecture, with T1 representing any supporting activity opened during the lecture, such as a videoconference or a chat session; the supporting task should be finished upon conclusion of the lecture.

Allen's *during* relation has also been adapted to the specification of two dependencies, but in this case related to how many times a task can be executed while the other is being executed. *T1 duringA T2* indicates that just one single execution of T1 can be performed during T2 period. *T1 duringB T2* indicates that T1 can be executed several times during T2. Consider T2 to be an online evaluation and T1 the student interaction. Dependency duringA models that student's answers cannot be reviewed, whereas duringB would allow any number of revisions during the evaluation period.

T1 overlapsA T2, as Allen's overlaps relation, expresses that T1 is a task starting before T2, with T2 starting before T1 conclusion and T1 finishing before T2 conclusion. Dependency *T1 overlapsB T2* relaxes the specification of which task should finish first.

From Allen's *before* relation, first interval finishes before the beginning of second interval, three different dependencies have been derived. From the coordination point of view, there are two possible interpretations. The first, expressed as *T1 beforeA T2*, has the meaning that task T1 can only be executed whether before task T2, i.e., upon the start of T2 execution of T1 is disabled. This might be the case in which T1 represents a student taking

a test and T2 the presentation of the answers. The other possible interpretation captures the concept of a prerequisite. In this case, the start of the second task is only enabled after the conclusion of the first task. This expresses the situation in which the beginning of T2 depends on resources that will be available only upon conclusion of T1. In terms of the dependencies in this framework, this is expressed by two constructs, depending on the multiplicity of the second task. The first is *T2 afterA T1*, in which each execution of T1 enables one execution of T2. *T2 afterB T1* means that the conclusion of one execution of task T1 enables multiple executions of task T2.

2.2. RESOURCE MANAGEMENT INTERDEPENDENCIES

Sharing, simultaneity, and volatility are the three basic constructs defined to complement the specification of collaborative tasks regarding the availability of resources.

Sharing represents the situation in which the use of a limited number of resources, expressed as a parameter, is required for the execution of several tasks. The dependency *sharing 1* denotes mutual exclusion, since any task using the resource blocks all other tasks requiring that resource. Consider an online lecture in which students want to ask questions: There might be many students "raising their hands", but only one would be able to ask the question at a time since only one resource (an audio connection to the lecturer) is available.

Simultaneity represents the situation in which the specified resource is only available when at least the number of tasks specified in the dependency request the resource. For example, dependency *simultaneity 2* could be used to represent that a chat session should not be opened if requested by only a single user.

Volatility indicates whether a resource, after is use, is still available for other tasks or is consumed by the task that has taken it. A parameter specifies how many times the resource can be used before it becomes unavailable to the system. For example, to specify that each group of students should work on a distinct assignment, the resource "pool of assignments" would have *volatility 1*.

These dependencies can be combined in order to express more complex relationships between tasks and resources. For example, sharing 2 volatility 3 express that a given resource can be used by up to two tasks simultaneously, but after its third use the resource becomes unavailable.

2.3. TIMEOUT MECHANISMS

Analysis of the framework constructs shows that they may be used in the coordination of collaborative tasks in two distinct situations. In the first form, interdependencies are used to specify activation of related tasks; e.g., *T1 startsB T2* indicates that T2 is activated as soon as T1 starts. This is characterized as an active interdependency. In the other form, a passive interdependency characterizes the situation in which the relationship between tasks is not of a task starting or forcing the execution of another, but that of restricting the collaborative activity to be performed only when involved tasks satisfy the dependencies. In this case, the same dependency *T1 startsB T2* indicates that even though T1 is ready to start its execution, its beginning has to be delayed until T2 is also ready.

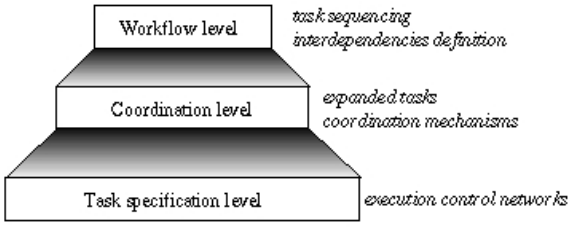
Another possible usage of passive interdependencies is in the validation of an activity, similar to the concept of a database transaction. It enables to express that an activity is complete only when all involved tasks, satisfying the specified dependencies, are finished.

Even though the interdependencies passive interpretation may lead to deadlocks, it represents conditions that cannot be disregarded. To deal with this situation, the framework defines two timeout constructs. The first type of timeout establishes alternative tasks to be performed when the specified waiting time expires. The second type brings the waiting tasks back to their initial states upon expiration of the waiting time. The usage of these constructs is illustrated in Raposo et al. [8].

3. PETRI NET MODELS FOR THE COORDINATION MECHANISMS

Mechanisms were developed to coordinate the above set of interdependencies using Petri nets [8]. The formal modeling enables a designer to anticipate and test the environment behavior, thus avoiding the usual trial-and-error approach. Petri nets have a well-established theory and can capture the main features of a collaborative environment, such as non-determinism, concurrency and synchronization of asynchronous processes. They also accommodate models at different abstraction levels and are amenable both to simulation and formal verification.

In this proposal, the design of a collaborative learning environment is divided into three hierarchical levels: Workflow, coordination, and execution (Fig. 1). In the workflow level, each participant's behavior is modeled separately, establishing the interdependencies between tasks assigned to the environment actors. The coordination level is built under the workflow level by expansion of interdependent tasks and insertion of the correspondent coordination mechanisms between them. The environment model is simulated and analyzed at this level. The execution level deals with the actual execution of tasks in the environment.



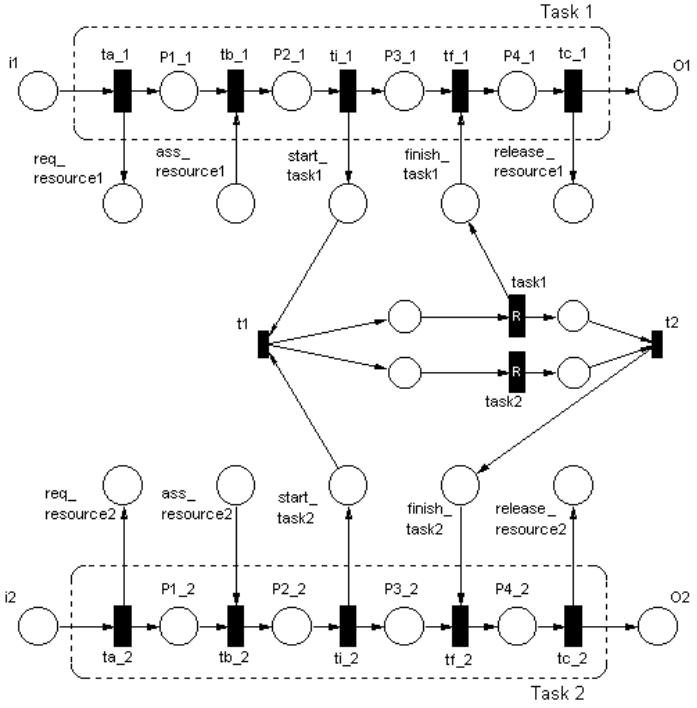
During the passage from workflow to coordination level, each task that has a dependency with another is modeled by a net with five transitions (ta, tb, ti, tf and tc) and four places (P1, P2, P3 and P4), as proposed by van der Aalst et al. [9]. Attached to each expanded task there are five places representing the interaction with external entities. The places request_resource, assigned_resource and release_ resource connect the task with the resource manager. The places start_task and finish_task connect the task with the temporal coordination mechanism and the agent that performs it, respectively indicating beginning and end of task execution.

Fig. 2 presents the model of the mechanism for relation Task 1 startsA Task 2. (The full set of models is available at http://www.dca.fee.unicamp.br/~alberto/pubs/IJCSSE/mechanisms.). Transition t1 ensures the simultaneous beginning of both tasks, and t2 ensures that Task 2 will finish only after the end of the other task. The transitions task1 and task2 represent the execution of the tasks. They are modeled by transitions with token reservation (indicated by R), which are non-instantaneous transitions; tokens are taken from their input places when they fire and only some time later, representing the duration of the tasks execution, they are added to their output places.

4. MODELING ACTIVITIES IN A COLLABORATIVE LEARNING ENVIRONMENT

The Sapiens Project aims to provide technological support for collaborative learning. The project brings together educators, engineers, and computer scientists, working in the definition of an environment framework that can be used according to an instructor or mentor needs.

One of the Sapiens applications is an environment to collaboratively construct Web documents. This activity can be summarized as follows. Instructors in a course define the subject and related issues to be discussed by the class. To provide background for the discussions, they also suggest recommended readings. The students work on the subject to produce a document contemplating the issues. This document production is performed in a cyclic process, with initial rounds generating preliminary versions and the last round producing the final document. In the workflow level, considering the case in which only one review is permitted, this relationship could be freely expressed as: Document production *happens afterA* Problem definition; and Document production *uses*



resource Versions *with volatility 2.*

Fig. 3 presents the corresponding coordination constructs with tasks expanded, omitting labels for individual places and transitions. Between the task problem definition, on top, and the task document production, below, the model for mechanism afterA is introduced. It guarantees that document production does not happen before the subject is defined. Below the document production task, the resource management mechanism to limit the number of generated versions is introduced. An inhibitor arc (with a circle at the arrowhead) indicates when the final version is produced, that is, when a document is produced and no more versions are allowed.

This figure presents only the top-level coordination view. Obviously, any task could be further refined in terms of subtasks, which in turn could be related by other coordination constructs. Consider the execution of the document production task (T1). Students have a period to individually read the assigned texts and build their personal standings about the proposed discussion issues. After this period, they work together in small groups, which have one member assigned as the group moderator. The moderator presents the group conclusions to the remaining of the class and integrates the group of editors for the document. Moderators also lead discussion sessions to settle common points and to detect controversial issues. Then they compose and publish a draft document contemplating the selected subject and issues. This draft becomes the assigned reading for the next round, again with individual reading, small groups discussion, and general sessions. The final document is the result of the editors' work after the final general session. From this overall presentation, a possible expansion for task T1 is derived:

T2: Read assigned text. As above, several subtasks compose this task: Get the text assigned in the problem definition task, read it, and take notes related to text parts (sections, paragraphs, phrases). Result of this task is a set of notes expressing doubts or the individual position on the subject.

T3: Discuss issues in small group. Present results of T2 and proceed to settle common points of agreement and detect controversial points. Result of this task is a list of issues to be taken to the general session.

T4: Discuss issues in general session. Moderators present summary of T3 results. From the lists of issues of T3, the class elects relevant discussion points, which are debated in general sessions to define the overall document structure, and points out relevant content to be present in the produced text.

T5: Edit document. The moderators/editors work in order to produce a document following the guidelines established during T4. The result is a hyperdocument published in the Web.

Several computer-based components were integrated to support these tasks. AnnotTool, a Web-based annotation tool, supports T2 [10]. T3 and T4 start in presential meetings and continue using AnnotTool and UseNet discussion lists. WebDAV and CVS supports execution of T5.

The set of coordination mechanisms can be again used here to define how these tasks should be performed. One possible approach would be to sequence these tasks execution: T3 afterA T2, T4 afterA T3, T5 afterA T4. Another less restrictive approach would allow text reading to continue during discussions, thus enabling the students to review their positions, until the moment that the document is produced (end of T5). Another possible variation is to express that the document starts to be produced with the small groups discussions (T3), thus

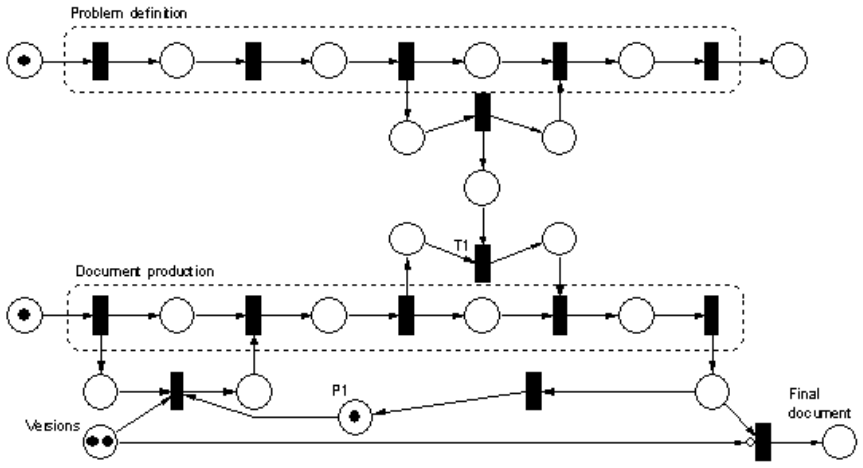


Figure 3: Relating problem definition and document production

yielding to T5 finishesA T2, T3 meets T4, and T3 startsA T5.

Once these interdependencies are established, the designer is able to evaluate the environment usage and define which tools should be enabled at any given moment.

5. CONCLUSIONS

The design of collaborative learning systems used to emphasize computer mediated communication tools, such as e-mail, bulletin boards, whiteboards, chat rooms, audioconferences, and videoconferences. Eventually, some of these tools were integrated to some content delivery mechanism. With the expansion of the Web, many systems have adopted the delivery of content codified in HTML through the HTTP. However, the coordinated usage of such tools were usually considered to be an activity performed outside the system, under some external supervision or by social agreement.

Recent work in CSCW considers coordination as integral part of the design of collaborative systems, and this issue cannot be disregarded in the design of computer supported collaborative learning systems. Different pedagogical approaches shall lead to the design of different CSCL systems, but each system should not be designed from scratch. Coordination mechanisms are going to be part of the glue that put software components together to assemble collaborative learning environments.

This paper presented a set of coordination mechanisms to express relationships between tasks. Considering that any activity can be decomposed as a set of interdependent tasks, such mechanisms provide a framework to design environments in a CSCW or CSCL system devoted to support that type of activity. The use of these mechanisms was illustrated in the specification of an environment to collaboratively construct Web documents.

Modeling tasks and their coordination dependencies with Petri nets enables to simulate and analyze the system before it is built. A simple translator from coordination constructs to Petri nets was developed, thus enabling the automatic generation of complete Petri nets modeling for an environment. Using Petri net simulators, it was possible to perform reachability and deadlock analyses for the system.

6. ACKNOWLEDGEMENT

FAPESP supported the Sapiens Project (97/128071) and Dr. Raposo stay at Unicamp.

REFERENCES

[1] K. Schmidt and L. J. Bannon, Taking CSCW seriously - Supporting articulation work. *Computer Supported Cooperative Work, 1*(2), 1992, 7-40.

[2] T. W. Malone and K. Crowston, What is coordination theory and how can it help design cooperative work systems? *Proc. Int. Conf. on Computer Supported Cooperative Work (CSCW)*, Los Angeles, USA, 1990, 357-370.

[3] T. W. Malone and K. Crowston, The interdisciplinary study of coordination. *ACM Computing Surveys, 26*(1), 1994, 97-119.

[4] W. Deiters, T. Goesmann and T. Löffeler, Flexibility in workflow management dimensions and solutions, *Int. J. Computer Systems Science and Engineering, 15*(5), 2000, 303-313.

[5] M. A. Vouk, D. L. Bitzer and R. L. Klevans, Workflow and end-user quality of service issues in Web-based education. *IEEE Trans. on Knowledge and Data Engineering, 11*(4), 1999, 673-687.

[6] 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. *Proc. 7th Int. Workshop on Groupware (CRIWG)*, Darmstadt, Germany, 2001, 170-179.

[7] J. F. Allen, Maintaining knowledge about temporal intervals, *Communications of the ACM, 26*(11), 1983, 832-843.

[8] 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 and Engineering, 15*(5), 2000, 315-326.

[9] W. M. P. van der Aalst, K. M. van Hee and G. J. Houben, Modelling and analysing workflow using a Petri-net based approach, *Proc. 2nd Workshop on Computer-Supported Cooperative Work, Petri nets and related formalisms*, Zaragoza, Spain, 1994, 31-50.

[10] C. M. Adriano, A. B. Raposo, I. L. M. Ricarte and L. P. Magalhães, Changing interaction paradigms in annotation environments, *Proc. World Conf. on Educational Multimedia, Hypermedia & Telecommunications (EDMEDIA)*, Montreal, Canada, 2000, 28-33.