Coordination in CE Systems: An Approach Based on the Management of Dependences between Activities

Sylvain Lizotte & Brahim Chaib-draa

Computer Science Department,

Laval University,

Ste-Foy, PQ, Canada, G1K 7P4

e-mail: {lizotte,chaib}@ift.ulaval.ca

Short form of the title: Coordination = Management of Dependences

Author to whom correspondence should be sent: B. Chaib-draa

Mailing address: Département Informatique, Pav. Pouliot,

Université Laval, Ste-Foy, PQ, Canada, G1K 7P4

Tel: (418) 656-2131 ext: 3226

Fax: (418) 656-2324

Email: chaib@ift.ulaval.ca

Coordination in CE Systems: An Approach Based on the Management of Dependences between Activities

Abstract

Coordination is a crucial problem in CE systems and it is neither easy to obtain nor to maintain. Our work is an attempt to develop a general model for coordination which can be adapted for some situations in the context of CE. For this purpose, the coordination definition developed by Malone [25] has been adopted. Coordination is then defined as the process of managing dependencies between activities. In this context, a theoretical model is presented that allows one to determine how to model an agent's activities and how to detect dependencies between those activities. In our model, major concepts are developed in terms of components of coordination, situations of coordination, coordination mechanisms and the coordination process. In this paper, we detail this model and then, we present an illustrative example and finally, we identify the current status and the future evolution of our approach.

Keywords: coordination, multiagent systems, management of dependences, distributed decision making.

1 Introduction

Like other engineering disciplines, concurrent engineering (CE) consists of a set of processes, methods, data, knowledge and tools to be used to improve productivity. The CE requirements can be broken down into the following categories:

1. *Team Work.* In CE, an interdisciplinary team is usually involved in a continuous cycle of specifying, scheduling, implementing, monitoring and improving the vital activities needed to the success of a given product. To this end, we should have the ability to i) define *agents* (person, process or software), ii) identify and distribute the activities needed to a given product, iii) identify and distribute the resources needed to the different activities, iv) plan and schedule the different activities resolving conflicts and favoring *concurrent activities* in order to improve productivity and quality gains.

2. *Coordination Between Agents.* Dependences and interdependences between agents' activities should be managed efficiently in order to avoid conflicts and favor concurrent activities. To achieve this, we should have the ability to i) identify dependencies between agents' activities; ii) capture the process of group decision making and define strategies for negotiation and conflict resolution; iii) elaborate communication between agents in order to support the group decision and negotiation.
3. *Focus on Quality, Time and Stock Reduction.* We generally consider CE as a multiagent environment wherein multiple team/agents interact closely, coordinate their respective activities by managing multiple dependencies in as timely a fashion as possible in order to *continually* improving: i) the quality control; ii) the cycle time reduction and, iii) the stock reduction. This is a major departure from the traditional engineering approach [30, 37].

This paper contributes to the second item, that is, the coordination between agents in the context of the concurrent engineering environment. In this type of environment, the design activities are highly interdependent. As mentioned by Klein [21], (notice that the discussion that follows is adapted from [21]) these interdependences need to be managed efficiently in order to avoid the possibility of frequent conflicts and favor concurrent activities. Currently, coordination is typically obtained through team meetings, notices, or centralized approach. In the team meetings approach, decision makers from different disciplines reason on each others' design decisions as a team. Such meetings require generally expensive resources (e.g. wasteful time, expensive collocation of the participants, etc.). Notices are recipes describing a design decision sent by a responsible to all the affected groups that might be affected. In this case, we can fail to identify *all* affected groups and we can also miss consistency. On the other hand, the affected groups are often overwhelmed and might misunderstand the sending memos. Finally, in the centralized approach, all the design decisions are gathered in one place and one decision maker reviews them for coordinating all tasks, resources and machines. Such an approach is often highly expensive and is not accurate enough. Furthermore, it is not generally updated frequently, otherwise the team would swamp the available bandwidth and the centralized decision maker would become a severe communication bottleneck.

The weaknesses of these classical approaches have had a critical impacts on production time and cost particularly in the CE organization. The model of coordination between agents proposed in this paper is a step to address those weaknesses. In our model, a CE organization is considered as a multi-agent system (MAS) composed of agents interacting more or less autonomously. These agents have decision and communication capacities, and common and/or individual goals (possibly conflicting). In that context, an agent is not acting alone on the environment. Agents act in the presence of other agents which, at the same time, make decisions and act in the environment. These interactions oblige agents to coordinate with each other. Unfortunately, coordination is neither easy to obtain nor to maintain. This is a crucial problem in MAS [14, 17, 18]. Generally, coordination has not been well understood and it has been examined from different angles by many researchers [26]. The importance and the need for developing a model for coordination is recognized by many researchers [9, 11, 14, 18, 26]. This type of model must be general and abstract enough so it can be used in several domains.

The purpose of this paper is to present an attempt to develop such a model that could be adapted to any domain. This paper is organized as follows. We begin in section 2 by presenting related work that have influenced our theory. We then present and motivate, in section 3, the adopted definition for coordination and see how coordination is related to collaboration in CE. Section 4 describes two types of coordination: the implicit and explicit coordination, thereafter, section 5 introduces our model of coordination. Section 6 details an illustrative example and section 7 gives the current status of the proposed model and present some possible future developments. Finally, section 8 concludes this paper.

2 Related Work

Nowadays, there are many approaches to CE based on multiagent systems [3, 22]. However, no one of them studied the dependences between agents, tasks and resources in the CE context. In this paper, we propose a model to address this shortfall. In fact, the study of coordination is not an exclusive subject of CE research. It is rather a subject at the intersection of several disciplines. Our theory is, of course, developed in a computer science context and more precisely in a multi-agent context. However, the insights that are provided by other fields of research are

important and can not be ignored. For instance, the work of Malone [25, 26], Crowston [11, 12] and Castelfranchi [5, 6], which brings principles and observations useful for the development of a theory of coordination and which are not done in a computer science context, have influenced our work. In this context, we have been precisely inspired by the coordination definition and the components of coordination developed by Malone and Crowston. This latter, did identify one principle on which our work is based: a coordination situation can be identified by analyzing dependencies between coordination components.

In a computer science context, the work of Decker [14] and von Martial [36] have also influenced our work. Indeed, our work exploits the task structure developed by Decker in TÆMS and the coordination relations identified in GPGP (generalized partial global planning). Finally, from von Martial's work, important insights have been gained, for instance, the positive (hard) and negative (soft) notion of coordination relations.

Work on coordination is not limited to these five pieces of work. There exists much more research that influences the development of a general coordination theory. For instance, alternative approach [18, 19], coordination mechanisms [23, 32], negotiation protocols [4], conventions managing commitments [17], mechanisms for coalition formation [33], and distributed planning [15]. Some of this work uses strong and restricting assumptions; others are more general, but also less effective.

In the CE context, Prasad [29] (pp. 220-225) has introduced seven components to the team cooperation: 1) collaboration; 2) commitments; 3) communication; 4) compromise; 5) consensus; 6) continuous improvement and 7) coordination. The author expressed the company's productivity as the sum of individual contributions and productivity gains due to team interactions, with the idea that interactions here may reinforce or nullify efforts of the interacting teams. If a magnitude of an interaction between teams is positive, it is called cooperation. If it is negative, it could be due to controversies or unconstructive competitions between the parties involved. Thus, the net contribution of all the working teams will depend on the signs of the interactions– whether the effort is in cooperation or in controversy. This means that in any CE organization, the efforts of all work-groups will be mixed: some additive and some substractive. Now how to lure positive interactions from teams/agents is what the CE management needs to explore.

Prasad suggested a) to develop methods to work toward the "self interest" of the teammates in such a way that the net outcome is in the best interest of both the teams/agents and the company; b) to involve members in mapping the decomposing tasks of the CE project/work and in identifying the time, concurrency, and resources required for each step. This work has influenced our research and more specifically our investigation of CE by techniques issued from the multiagent systems. Thus, the model which is presented in this paper focuses on the previous second aspect (i.e., b))since we introduce here a theoretical model for the coordination which is based on how to manage dependencies between agents, activities and resources. By this way, we involve members of the CE project in the decomposing tasks and identifying the dependencies between the sub-tasks as proposed by Prasad.

In CE also, Klein has developed the iDCSS system [20]. In this system, cooperative product development is distributed across the three orthogonal axes of agents, perspectives and time. Distribution across agents requires support for the sequencing of tasks and flow of information among participants; this is addressed by process management technologies (i.e., workflow, planning/scheduling). Distribution across perspectives requires to maintain consistency among agents' design actions since these agents work on different interacting aspects of the design and/or have different often-incompatible goals; this consistency is maintained with the support of conflict management technologies. Finally, distribution across time requires latter on the availability of rationale for design decisions made at an earlier stage; this availability is supported by the memory management technologies. This work has some similitudes with our approach. Indeed, the process management and conflict management of Klein's model are similar to the planning process and the selection of the appropriate mechanism of coordination, two important steps in our approach. Klein's model seems however more appropriate for product development, whereas our approach is abstract enough so it can be adapted to different domains.

Recently, the Concurrent Engineering Research and Applications (CERA) journal has issued a special issue devoted to "The Application of Multiagent Systems to Concurrent Engineering" [3]. The first article [1] addresses the issues of design, manufacturability analysis, incremental process planning, dynamic routing and scheduling simultaneously. A fine-grained multiagent architecture that has been developed to support this level of CE is presented. The second

article [2] looks at conflict among agents, one the primary impediments to effective concurrent engineering. It presents a model of conflict management between single functions (at the very fine level of granularity) considered as agents. By considering such a very fine level, authors brings to the surface many of the "fine" conflicts that are generally hidden in a coarse structure. The third article [10] discusses the design of virtual product development environment to support the CE process. It provides a vision of tools, techniques and processes that are intended to immerse a designer in an interactive simulation environment. The article describes the range of services that are required, and discusses what technologies are currently available and what is still be developed. The fourth article [13] is concerned with relationships between decision makers. It describes precisely, how the preferences and constraints of the supervisor can be distributed to the subordinates (using contracts), so that they can use their own knowledge, know-how and expertise. Subordinates are free to apply their own preferences as long as they do not violate contracted preferences. They interact to solve a problem by means of distributed constraint satisfaction, during which search heuristics attempt to identify the best design. In the fifth article [24], Lander and her co-workers describes some agent-based approaches in the context of blackboard technology and presents results from a robust industrial application. In the sixth paper, Peters and Sohi [27] propose an Object Petri-Net (OPN) model for coordination of multiagent systems based on fuzzy clocks which takes a refined design-to-time approach in real-time scheduling of CE projects. Agents tasks here are subdivided into required and optional subtasks to take advantage of the imprecise nature of the computation performed by an agent. Concurrent with each agent is a fuzzy clock which measures the timeliness of an agent-task relative to temporal intervals and prescribed deadlines. In the context of a CE project, agents cooperate with a coordinator in managing their time, and in periodically supplying tasking status reports for a blackboard. With this approach, we can make appropriate adjustments in the overall schedule whenever agents fail, disappear, or are in conflict. Finally, Sobolewski's article [35] describes a way of providing system integration by using a multiagent knowledge-based environment that encompasses programs, tools, and knowledge bases.

Thus, the intersection of multiagent systems (MAS) and CE, although still quite new, is becoming increasingly active. Another point of note is that the coordination seems to be the

cornerstone of CE. However, no approach did investigate a general model for the coordination that could be adapted to any domain or situation. The purpose of this paper is to present an attempt to develop such a model. Before to give details about this model, we need to adopt a definition of coordination and elaborate more on collaboration in CE.

3 Coordination and Collaboration in CE

A definition of coordination should be based on the reasons that justify the need for coordination. In multiagent environments Jennings [17] has identified three reasons justifying the need for coordination: 1) there are dependencies between agents' actions; 2) there is a need to meet global constraints; 3) generally, no one agent has sufficient competence, resources or information to solve the entire problem.

These three reasons identify three cases of interactions that coordination tries to manage. These three interactions can all be represented by dependencies between agents' activities. The first case is obvious since actions are dependent. In the second case, agents must satisfy global constraints, their activities are hence directed towards common goals. The agents' activities are then dependent due to the existence of those common goals. Finally in the third case, one agent does not possess all the capacities, resources or information needed to resolve a problem. In other words, each agent is depending on the competence of other agents and hence on their activities. These considerations show clearly that coordination consists in *managing dependencies between agents' activities*. This definition which is the most cited definition of coordination [8, 14, 17, 18, 28, 29] is by MIT Sloan School of Management [26]. It expresses that coordination involves actors performing interdependent activities that achieve tasks, and its analysis include task decomposition, resource allocation, synchronization, group decision making, communication, and the preparation of common objectives. Thus, such a definition is consistent with the fact that if no dependencies exist then nothing needs to be coordinated.

In the light of this definition, we can see CE as a collaborative activity wherein multiple entities interact closely by coordinating their respective activities (resolving thus conflicts and favoring concurrent activities) in order to achieve a better product. Thus, coordination is necessary to collaboration between entities (teams, computers and processes) in CE organizations.

Without such a collaboration, the CE community may quickly degenerate into a collection of chaotic, incohesive individuals. Table 3 shows various forms of knowledge sharing and collaboration between concurrent teams, computers, and processes as discussed in [29].

Table 1: Forms of Sharing and Collaboration (after [29]).

Modes	**Team**	**Computer**	**Process**
Team	e.g., Help Desk, Customer Support, Intra and Inter-teams Cooperation, etc.	e.g., User Interface, Information Retrieval, Data Inputs, etc.	e.g., Interactive Modeling, Analysis, CAD-CAM, CAE, CAD/Conferencing, etc.
Computer	e.g., Display, Online Help, Desktop Applications, Plots, etc.	e.g., Inter computer, LAN, Internet, email messaging, Network, etc.	e.g., Concurrent multiple processes, DNC-Postprocessing, Concurrent Sessions, etc.
Process	e.g., Software Agents, System Adm., Network Adm., Groupware, etc.	e.g., Feedback, Trouble Shooting, Remote Diagnostics, Fault Mgt, Process Control, etc.	e.g., Inter-process, Groupware, Remote Applications, X-windows Applications, Multimedia, etc.

As precised by Prasad (the discussion that follows is adapted from [29] (pp. 302-303)), the age of our open world based on the competitive advantage requires to maximize the effectiveness of collaboration in any CE organization. The achievement of this is generally based on coordination of activities and sharing of responsability. On one hand, coordination of activities allow work-groups[1] (working in teams) 1) to do some activities in parallel; 2) to synchronize some other activities; 3) to use resources efficiently; 4) to share results at the right time; etc. All these aspects contribute to the overall objective of CE which consists of achieving a good product in the required time.

On the other hand, the sharing of responsability between teams, computers and process allows

[1] A work-group is a collection of sub-teams or people having in common interests or expertise and usually has a short life in a CE organization.

to maximize the effectiveness of cooperation and distributed decision making between these entities. Thus, one can shift responsability from human to the intelligent agents (computers) for routine or repetitive tasks. Some other examples include delegating responsability, reporting, evaluating results, etc. In general, the adopted strategies of communication and coordination determine the actions and basis of how to distribute responsability between intervening parties in CE organization.

To sum up, we have defined coordination and explained why this notion is the cornerstone of CE through the collaboration between teams, computers and processes. In fact, there exists two sorts of coordination: explicit and implicit. The next section presents these two sorts through the processes underlying coordination.

4 Implicit and Explicit Coordination

The development of a model of coordination involves understanding the processes underlying coordination. According to Malone [25], the coordination process depends on group decision-making, communication and perception of common objects. The coordination process involves making and accepting a decision by a group of agents (for example, what is the best allocation of tasks?). This process is generally referred to as *group decision-making* in the sense that several agents must determine the best choice that ensures coordination.

Evidently, group decision-making requires that the group members communicate with each other to determine the best choice. Communication also allows agents to update their knowledge, particularly, the required knowledge for coordination. Finally, the process of communication depends on the agents' capacity to perceive the same objects. The same objects include for instance: shared variables, blackboards, common knowledge, shared situations, shared databases, shared contexts, etc. If agents do not have this capacity, ambiguities can occur in communication between agents. Generally, those ambiguities cannot be solved without appealing features of agents' shared situations.

There exists two approaches to coordination: implicit and explicit coordination [7, 16]. Implicit coordination is generally based on a priori reasoning. More precisely, this type of coordination is possible if the CE designers can establish behavioral rules (during the specification

step) that agents are going to respect, in order to act in a coordinated manner. These rules are generally called *social rules* [34]. The coordination process is called *implicit* because agents choose only alternatives that respect the social rules. For instance, implicit coordination between two vehicles (agents) that are going in opposite direction on a two-way road, can be obtained by applying the following social rule: "An agent always drives on the right side of the road". This rule avoids frontal collisions. Explicit coordination, on the other hand, is done dynamically and necessitates that agents make explicit choices to ensure coordination. In order to do such explicit choices, agents need to possess reasoning capacities on potential interactions between their activities and those of other agents. In case of conflicts, agents generally use voting, negotiation (consensus) or the intervention of an authority. Explicit coordination then involves identifying coordination situations and managing those situations dynamically.

Table 2 presents a comparison of implicit and explicit coordination using the processes underlying coordination. Implicit coordination is generally easier to design (less costly) and more effective than explicit coordination because two of the processes underlying coordination (processes 2 & 3 of table 2) are reduced to a minimum. Thus, the decision-making process is determined in advance using conventions called "social rules". In this way, any agent has facilities to coordinate her activities with other agents, and negotiation (and more generally communication) is requested only when necessary. In the same context, communication between agents is done by exchanging signs and signals[2]. *Signs* can be reviewed as data representing a state in the environment with reference to certain norms for acts. The communication based on signs is generally as follows: a sender makes a sign which refers to some state of environment and which has the end of signifying, or letting the receiver knows the same reference. Of course, the sender and the receiver should share a set of signs with their references in order to communicate efficiently. *Signals* are another kind of information that agents can exchange. They can be considered as data representing time-space variables from a dynamic, spatial configuration in the environment and they can be processed directly by the agents as continuous variables. In communication by signals, the signal delivered by an agent i has the end of simply eliciting a reaction by j.

In the case of explicit coordination, group decision-making and communication are time

[2] The communication by exchanges of sign, signal and symbol is described in [7].

costly because they are more complex. This is due to the need for dynamically making common decisions between agents.

Implicit coordination reduces the autonomy of an agent because an agent must only consider alternatives with respect to some social rules. Using social rules can then, in some cases, reduce an agent's activities. Moreover, social rules are strongly domain dependent. Thus, the social rules about driving, stated previously, are good for an American agent, but an agent conforming to these rules in England will certainly cause accidents. Following the same idea, social rules oblige designers to anticipate every possible situation a priori. It can be very difficult to do this when the uncertainty about the possible interactions is high.

On the other hand, explicit coordination increases an agent's autonomy since the agent possesses a maneuvering margin that is not limited to alternatives respecting social rules. Moreover, increasing an agent's autonomy implies dealing with information uncertainty about the environment.

Each agent in a multi-agent environment must be able to use both types of coordination (implicit and explicit) depending on the situation that the agent has to consider. Indeed, in a real environment agents are generally confronted by three types of situations: routine, familiar and unfamiliar situations [7, 31]. Routine situations are situations where an agent reacts to a stimulus in her environment. It is a kind of stimulus-response behavior motivated generally by social rules. Familiar situations are situations where agents can generally coordinate their behaviors using some predetermined patterns based on expectations of future actions of other agents' actions and beliefs. Finally, unfamiliar situations are situations where it is difficult to obtain and maintain coordination, since agents need to be constantly informed of all developments in order to elaborate their decisions. Signals and signs, previously introduced, invoke routine and familiar situations respectively. Precisely, in routine situations, agents rely on signals to interact efficiently. In familiar situation they use signs to communicate between them. In unfamiliar situations, agents generally communicate by signs or symbols. *Symbols* are abstract constructs related to and defined by a formal structure of relations and processes which according to some conventions can be related to features of the external world.

Notice that an environment does not generally consist of purely routine, familiar or unfa-

Table 2: Explicit and implicit coordination regarding processes underlying coordination.

Processes	**Implicit coordination**	**Explicit coordination**
1. Coordination	less costly to design and more effective but can reduce agent's autonomy and it is domain dependent.	more costly to design and less effective but can improve agent's autonomy.
2. Group decision-making	static, i.e., the decision is determined in advance using conventions (social rules).	dynamic, i.e., implies evaluating alternatives and making a common decision between agents by consensus (negotiation), by vote or by authority (decision power).
3. Communication	generally done by exchanging signs and signals.	generally done by exchanging alternatives, evaluations and choices (symbols).
4. Perception of common objects	perceiving same agents and objects (sharing data files, blackboard, etc.).	knowing part of agents' intentions, goals and plans.

miliar situations. These three kinds of situations are present at different instances of time. In routines, coordination is implicit because interactions are predictable and the management of these situations can be determined partially or completely a priori. In familiar situations, coordination can be implicit or explicit (or a mixture of both) depending on the existence of predefined rules for the coordination in those situations. In unfamiliar situations there are no adequate rules to ensure coordination insofar as interactions are uncertain and dynamic. In this type of situation, the only way is the explicit coordination ensured by agents which are capable to reason about their potential interactions and negotiate with one another as needed. In this case, it is convenient to design 1) agents that are able to reason explicitly about other agents; 2) agents such as each agent must be able to evaluate the consequences of her decisions on others and the consequences of others' decisions on her goals and actions.

5 Model of Coordination

In this paper, we examine an approach for the implicit coordination. We achieve this in a simulated environment where autonomous agents manage their activities according to the available resources. These autonomous agents try to find the best way to coordinate their activities in the simulated environment and in the sense, they use the explicit coordination. Rules of coordination tested and validated in the simulated environment constitute "local" rules of behavior that lead agents to act in apparently coordinated way in the real environment. Thus, we use explicit coordination in a simulated environment to determinate rules for the implicit coordination in the real environment.

To present our model, the next section first defines what we called dependencies. Once this is done, section 5.2 will present how our theory tries to manage those dependencies.

5.1 Dependencies

Malone has presented four coordination components used to analyze coordination [25]: agents, resources, tasks and dependencies. In our model, agents are defined as entities capable of executing tasks and capable of making decisions on the execution of their tasks. Resources are objects in the environment (including information) that are used and/or created by tasks. A

task is work to be done, i.e., a sequence of actions executed by an agent toward achieving a goal. This work can be physical like "lifting a block", or cognitive like "processing a piece of information". Finally, dependencies are presented as components but their nature is different than the three other components. If a component x depends on a component y, then there exists directed relation from x to y. The meaning of this relation hinges on the components involved. It can mean, in the case of x and y being agents, that x is under the authority of y; in the case of x and y being resources, that x is a part of y; or, generally, that x can not be realized without the action or intervention of y[3]. Three attributes can characterize dependencies:

Unidirectional/Bidirectional: An unidirectional dependence of a component x on a component y means that x depends on y but y is independent of x. Unidirectional dependencies are simply referred to by *dependence*. On the other hand, bidirectional dependencies are referred to as *interdependence*, meaning that x depends on y and y depends reciprocally on x. Moreover, interdependence usually exists between components of the same nature (agent-agent, resource-resource or task-task).

Explicit/Implicit: An explicit dependence is generally known by an agent (this knowledge comes from communication or from the design of the MAS). Explicit dependencies are useful for detecting coordination situations and inferring implicit dependencies. Implicit dependencies are hence dependencies inferred from explicit dependencies. For example, a dependence of a task on a resource is usually known (resources needed by the task), but the interdependence between two tasks using a common resource must be inferred.

Hard/Soft: Hard dependencies are dependencies that must be considered and solved in order to achieve an agent's activities (i.e., necessary). For example, a hard dependence of task T_1 on task T_2 can mean that T_2 must be achieved before starting to execute T_1. Soft dependencies are useful but not necessary. For example, a dependence of task T_1 on resource r is soft if the utilization of r improves the execution of T_1, but not using r does not inhibit T_1 from being achieved.

[3] In some cases, y can only improve the progression or the resulting state of x.

Table 3 presents explicit dependencies that can exist between agents, resources and tasks. Each cell represents a type of dependence that is a directed relation. For example, dependence type 3 represents the dependence of an agent on a task. Precisely, each cell answers the question: "How component x can depend on component y?". Thus for example, cell 3 answers the question "How agent a_x can depend on task T_j".

Table 3: Typology of explicit dependencies between two components.

	Agent (a_2)	**Resource** (r_2)	**Task** (T_2)
Agent (a_1)	(1) a_1 depends on a_2 if a_2 has authority on a_1; or if a_2 possesses a commitment toward a_1	(2) a_1 depends on r_2 if the possession of r_2 is necessary for a_1; or if a_1 must manage r_2	(3) a_1 depends on T_2 if the execution of T_2 is necessary for a_1
Resource (r_1)	(4) r_1 depends on a_2 if a_2 possesses r_1; or, if r_1 is managed by a_2	(5) r_1 depends on r_2 if r_1 is linked physically or logically to r_2	(6) r_1 depends on T_2 if the existence of r_1 hinge on the achievement of T_2 (i.e. T_2 create r_1)
Task (T_1)	(7) T_1 depends on a_2 if a_2 possesses the capacity of executing T_1 (i.e. the achievement of T_1 depend on the effort of a_2)	(8) T_1 depends on r_2 if using r_2 is necessary for executing T_1 (i.e. T_1 wouldn't be achieve without using r_2)	(9) T_1 depends on T_2 if T_2 is a sub-task of T_1; or if there exists a domain dependent relation between the two tasks

The dependencies presented in table 3 are hard. Soft dependencies can be obtained in a similar table in which terms as "necessary", "hinge on", etc. are substituted (implicitly or explicitly) by "useful", "facilitate", etc. Moreover, cells 2-4, 3-7 and 6-8 of table 3 are complementary from a coordination viewpoint. For instance, if agent a_x wants to execute task T_j (agent-task

dependence (3)) and T_j can be executed by a_x (task-agent dependence (7)) then if a_x has time, it can execute T_j and there is no coordination problem relative to the task execution capacity. It is also important to note that the relationships presented in this paper are those that we judge interesting for modeling coordination. Of course, other relationships exist between agents, resources and tasks but, at this stage, they have not been judged interesting in our approach. We can also see how table 3 includess Prasad's approach on possible relationships between pairs of tasks as precised in [29] (pp. 182-183) in the CE context. According to this author there are four possible ways for the possible relationships between pairs of tasks: dependent tasks, semi-independent tasks, independent tasks and interdependent tasks. A pair of tasks is said to be dependent if a task requires information, which is an output from another task. This type of relation is precised in table 3 by (9) under the vocable "domain dependent relation between the two tasks". Similarly, a pair of dependent tasks is said to be semi-independent if the transfer of output from one to the other is only a partial transfer. This relation is precised in table 3 by also: "domain dependent relation between the two tasks". Independent tasks has no representation since we do not need them for coordination, and interdependent tasks (T_i and T_j) mean that task T_i is in relation with T_j and vice-versa. Thus, the dependence notion presented here, between agents, tasks and resources, is more general and can be adapted to any application for studying the coordination. Now the question is: how can agents manage these dependences? The next section deals with this point.

5.2 Management of Dependencies

Our first objective is to represent an agent's activities in order to manage dependencies between different activities. To do this, we should adopt a representation of the two components: tasks and resources. The component task is represented by task structures based on TÆMS [14]. In TÆMS, one distinguish two levels: the objective and the subjective levels. The *objective* level describes the essential, 'real' task structure of an episode (a particular problem-solving situation). The *subjective* level describes the agent's local view of the objective problem-solving situation over time (e.g., how much does an agent know about what is really going on, and how much does it cost to find out). This subjective view is essential for evaluating control algorithms, because

agents must make decisions with only subjective information about the current episode. In this context of two levels, the objective task structure of a problem-solving episode can be described as follows: an episode E consists of a set of *task groups* T; $E = \langle T_1, T_1, \ldots, T_n \rangle$. Each task group has an arrival time $End(T_i)$, and a deadline $D(T_i)$. Each T_i is represented by a directed acyclic graph. The nodes of the graph are called *tasks* `T`. The root task is denoted by a task which is usually represented by the entire task group T. Tasks that have no children, but that are not the root task, are called *subtasks*. Tasks that have no children, are called *methods* noted M.

The main difference between our model and TÆMS model comes from our representation of resources. Resources are modeled using six attributes: shareability, exhaustibility, transferability, volatility, quality and cost. Another difference comes from our modeling of the utilization and the creation of resources that can be considered as hard or soft dependencies.

The existence of dependencies between agents, tasks and resources does not mean that there exists a situation that needs to be coordinated. If a situation exists where coordination is required then it is called a *coordination situation*. Generally, different situations of coordination can be identified based on the different dependencies that exist between components and some conditions characterizing those situations. For example, there exists a coordination situation if under some conditions a task can't use a resource because of the existence of other tasks. For example, in a flexible workshop, if a tool can be used by two tasks executed by two different agents, these two tasks are then dependent regarding the utilization of the tool, and, there exists a coordination situation if these two tasks need to use the tool at the same time and this tool can only be used by one task at the same time (shareability of the tool). For identifying these situations, it is necessary to determine how tasks executed by different agents can be dependent. In MAS, a task can be achieved by one or more agents using and creating one or more resources, these possible dependencies can be represented by different types of situation for coordination as follows:

1. tasks access the same resources (utilization and creation, i.e., task-resource and resource-task dependencies, i.e., cells 8 and 6 in table 3);
2. tasks possess explicit dependencies, generally domain dependent (task-task dependencies, i.e., cell 9 in table 3);

3. tasks can only be executed by some agents, i.e., different capabilities; or, the same task can be executed by different agents (redundancy), i.e., same capabilities (agent-task and task-agent dependencies, i.e., cells 3 and 7 in table 3).

Each of these types of situations can be a coordination problem or a coordination opportunity. A *coordination problem* is a situation that must be resolved in order to accomplish a task successfully. A such problem is generally detected thanks to hard dependencies. A *coordination opportunity*, on the other hand, is a situation that can be exploited to gain at the task execution level, and this opportunity is generally detected thanks to soft dependencies. Once a coordination situation is detected, agents must determine which coordination mechanism must be used for solving the problem or exploiting the opportunity. A *coordination mechanism* is a method followed by all agents (for example, negotiation protocols) that allows to process a situation of coordination. To process a situation of coordination, agents propose different solutions and adopt the pertinent one by using voting, negotiation (consensus) or the intervention of an authority. Once a solution regarding a coordination situation is adopted, there exists generally an agreement between agents involved. The chosen mechanism then allows agents to set commitments in order to ensure that they are going to respect the established agreement.

Finally, in an MAS, if a dependence exists between the activities of two agents, then one of those agents must inevitably detect this dependence in order to coordinate her behavior[4] with the other agents. Before executing a task, an agent must verify the existence of a coordination problem or opportunity between her tasks and others' anticipated tasks. To do this, some information must be exchanged between agents. The *coordination process* can be defined as the dynamic execution of some steps (for instance, the exchange of information) needed to detect and manage coordination situations. More specifically, the coordination process consists in identifying coordination situations and applying the corresponding coordination mechanism in order to resolve problematic interactions while taking advantage of cooperation opportunities. The results of the coordination process is a series of commitments between agents that ensure and improve the execution of agents' tasks. A coordination process generally includes :

[4] In centralized social organization, the agent managing dependencies is generally called the coordinator, i.e., the agent having the responsibility of coordinating the others.

1. PLANNING: each agent plans her activities;
2. DETECTION OF SITUATIONS OF COORDINATION: agents communicate between them their (partial) plans in order to detect situation(s) of coordination ;
3. DISTRIBUTED DECISION MAKING: the group selects the appropriate mechanisms of coordination to coordinate the partial plans;
4. COMMITMENTS: as result of the distributed decision making, agents are committed to do new (partial) plans;
5. EXECUTION: agents try to achieve their respective plans in order to respect their commitments;
6. MANAGEMENT OF COMMITMENTS: agents communicate their partial results with respect to their commitments. They have also to manage the unrespected commitments.

6 An Illustrative Example

We consider an example where two machines controlled by intelligent softwares (in this case, we have agents A and B where agent = machine + intelligent software) are required to place some piece of metal in a lathe, one (i.e., A) making a bolt and the other (i.e., B) a nut. Obviously, only one agent can use the lathe resource (R) at a time. We assume that such activity is an unfamiliar one for A and B which have other routine and familiar activities that are not considered here. The actions that A and B need to do in our example are (with $i \in \{A, B\}$):

1. m_i : agent i moves to the lathe;
2. p_i : agent i places the piece of metal in lathe;
3. b_i : agent i makes a bolt;
4. n_i : agent i makes a nut;
5. f_i : agent i returns from the lathe.

Agents A and B are considered here as autonomous agents having planning, decision-making, acting and communication capacities (in the current version, the communication capacities are based on interagent exchange using mailboxes). Furthermore, A and B are assumed sharing knowledge about their activities and about the model of coordination presented in this paper.

In this case, A and B can interact in order to coordinate their actions. To do this, they have to 1) plan; 2) detect situations of coordination, 3) make decision(s); 4) be committed to replan if necessary; 5) achieve their plans. A and B are further assumed to know the preconditions, the during condition and the postconditions. In these conditions, agents A and B can produce the following plans:

A: $m_A \longrightarrow p_A \longrightarrow b_A \longrightarrow f_A$

B: $m_B \longrightarrow p_B \longrightarrow n_B \longrightarrow f_B$

If we formally provide details on the components of coordination, i.e., actions, agents and resource, we can give an analysis of explicit dependencies between each pairwise of these three components as indicated in table 2. For our illustrative example, we only give sufficient details behind this analyze. Thus, dependence type (6), (7), (8) and (9) (see table 2) allow agents to establish the following situations of coordination:

1. b_A and n_B conflict one another because they share the same non-shareable (and non-consumable) resource R;
2. b_A has precedence[5] over p_B, but not vice versa;
3. n_B has precedence over p_A, but not vice versa;
4. p_A and p_B each have precedence over the other, but cannot be executed in parallel.

Let x_A^- and x_A^+ denote respectively the beginning and ending of the action x done by A. Let also the notation (x_A^-, R, x_A^+) denotes the sequence that agent A plans to do with the resource R. In these conditions, agents A and B make the following decision in order to meet some of the previous constraints:

A: $(p_A^-, R, p_A^+) \longrightarrow (b_A^-, R, b_A^+)$

B: $(p_B^-, R, p_B^+) \longrightarrow (n_B^-, R, n_B^+)$

A and B exchange those decisions and detect the following critical region:

[5] An action a has precedence on another action b if, and only if, a can be executed while b is suspended. Notice that if $a^+ \leq b^-$ the precedence constraint between a and b is called "temporal precedence".

(p_A^-, R, b_A^+) conflicts with (p_B^-, R, n_B^+)

Now agents A and B have to select an appropriate mechanism (or algorithm) to solve this synchronization problem. This synchronization problem is precisely a critical-section problem for which semaphores can be

implemented efficiently in order to solve it. If we give our agents the opportunity to use interagent-communication scheme using mailboxes, they can agree on the following coordinated plans:

1. solution for agent A:
   $m_A \longrightarrow [(B?f_B^+)_A^-, mbox_A, (B?f_B^+)_A^+] \longrightarrow [(B!p_A^-)_A^-, mbox_B, (B!p_A^-)_A^+] \longrightarrow (p_A^-, R, b_A^+)$
   $\longrightarrow [(B!f_A^+)_A^-, mbox_B, (B!f_A^+)_A^+]$
2. solution for agent B:
   $m_B \longrightarrow [(A?f_A^+)_B^-, mbox_B, (A?f_A^+)_B^+] \longrightarrow [(A!p_B^-)_B^-, mbox_A, (A!p_B^-)_B^+] \longrightarrow (p_B^-, R, n_B^+)$
   $\longrightarrow [(A!f_B^+)_B^-, mbox_A, (A!f_B^+)_B^-]$

In this scheme, each agent is assumed to have her own mailbox. In this case, the interagent-communication is the following: when the agent i *send* a message s to j she puts it in the mailbox of j; and when i receives something, she receives it in her mailbox. More precisely, The operation "i sends s to j" can be noted $[(j!s)_i^-, mbox_j, (j!s)_i^+)$, and the operation "$i$ receives s from j" is $[(j?s)_i^-, mbox_i, (j?s)_i^+)]$.

Notice that the synchronization is not done by a central synchronizer but by the two agents A and B which adopt a mechanism of coordination according to their capacities and knowledge about the situation. The plans that they are committed to achieve allow any interleaving or parallel execution that does not lead to deadlock. If we suppose now that this example reflects a routine activity, that is, a repetitive activity, then the same coordination between A and B is not done by agents themselves but by the designer. Coordination in this situation, is generally ensured by a fix algorithm which solves effectively the critical-section problem. In this case, agents do not plan, do not make a decision, they only execute the same fix algorithm. They behave like *reactive agents* for which the coordination is implicit.

7 Current Status and Future Work

We are currently soliciting actual product examples from a varied group of companies (automotive, robotic systems, and computer industries). By end 1998, we expect to complete a software prototype for coordination according to the model presented in this paper.

Currently, our work does not exploit the resource-agent and agent-resource dependencies because it doesn't seem useful, at this stage, to distinguish between the utilization and the possession of a resource. For example, to use a printer (resource), an agent must take control (logical possession) of the printer in order to print (task) a complete document. However, we can not discard these types of dependence without further study.

Future work will describe in more detail the coordination components, situations, mechanisms and processes introduced in section 5. Firstly, coordination components will be described by introducing the representation used to model them. Secondly, coordination situations will be modeled to explain how an agent can detect these situations. Thirdly, coordination mechanisms will be categorized and then linked to coordination situations. Finally, we will present the coordination process by algorithms defining the steps needed to detect and manage coordination situations using coordination mechanisms. Long-term work included the validation of our model with a real world implementation.

8 Conclusion

This paper has presented a model for coordination which uses dependencies between agents, tasks and resources. This model is very general and it can be adapted to several domains including CE activities. For this model, we have explained why the Malone's definition of coordination has been choose to develop it. We have discussed why it is important to use implicit as well as explicit coordination. Then, we have presented a topology of dependencies between components of coordination: agents, resources and tasks and explained how to manage these dependencies in order to obtain and maintain coordination between agents. We have also presented an example which illustrates the applicability of the proposed model. Finally, we have given the current status of the proposed model and presented some possible future developments.

Acknowledgments

This research has been financed by the Natural Sciences and Engineering Research Council (grant OGP–0121634 and scholarship #167735). We specially like to thank Clifford Grossner for reading and commenting on this paper.

References

[1] Balasubramanian, B. and D.H. Norrie, 1996. "A Multiagent Architecture for Concurrent Design, Process Planning, Routing, and Scheduling," *Concurrent Engineering: Research and Applications,* Special Issue on The Application of Multiagent Systems to Concurrent Engineering, 4, pp. 7-16.

[2] Berker, I. and D. C. Brown, 1996. "Conflicts and Negotiation in Single Function Agent Based Design Systems," *Concurrent Engineering: Research and Applications,* Special Issue on The Application of Multiagent Systems to Concurrent Engineering, 4, pp. 17-33.

[3] Brown, D. C., S. E. Lander, C. J. Petrie, eds. 1996. *Concurrent Engineering: Research and Applications,* Special Issue on The Application of Multiagent Systems to Concurrent Engineering, 4.

[4] Bussmann, S. and J. Müller, 1992. "A Negotiation Framework for Cooperation," *Proceedings 1992 of the Special Interest Group on Cooperating Knowledge Based Systems (CKBS-SIG 1992)*, Dake Center, University of Keele, pp. 1-17.

[5] Castelfranchi, C., M. Micely, and A. Cesta, 1992. "Dependence Relations among Autonomous Agents," in *Decentralized A.I. 3*, Elsevier Science Publishers B.V., pp. 215-227.

[6] Castelfranchi, C. 1995. "Commitments: From Individual Intentions to Groups and Organizations," *Proceedings of the First International Conference on Multi-Agent Systems*, AAAI Press/The MIT Press, Menlo Park, California, pp. 41-48.

[7] Chaib-draa, B. 1996. "Interaction between Agents in Routine, Familiar and Unfamiliar Situations," in *International Journal of Intelligent & Cooperative Information Systems*, 5(1), pp. 1-25.

[8] Chaib-draa, B., J. Desharnais, and S. Lizotte, 1995. *A Cognitive Map Formulation for Relationships between Agents*, Rapport interne du Département d'Informatique, DIUL-RR-9505, Université Laval, Ste-Foy, PQ, Canada.

[9] Cohen, P.R. 1991. "A Survey of the Eighth National Conference on Artificial Intelligence: Pulling together or pulling apart?" *AI Magazine*, 12(1), pp. 16-41.

[10] Cox, A., E. Bouchard, and D.H. Norrie, 1996. "SBD System Design," *Concurrent Engineering: Research and Applications,* Special Issue on The Application of Multiagent Systems to Concurrent Engineering, 4, pp. 35-46.

[11] Crowston, K.G. 1991. *Towards a Coordination Cookbook: Recipes for Multi-Agent Action*, MIT, Cambridge, Mass., CCS TR#128, Sloan WP# 3416-91.

[12] Crowston, K.G. 1994. *A Taxonomy of Organizational Dependencies and Coordination Mechanisms*, MIT, Cambridge, Mass., CCS TR#174, Sloan WP# 3718-94.

[13] D'Ambrosio, J., T. Darr, and W. Birmingham, 1996. "Hierarchical Concurrent Engineering in a Multiagent Framework," *Concurrent Engineering: Research and Applications,* Special Issue on The Application of Multiagent Systems to Concurrent Engineering, 4, pp. 47-57.

[14] Decker, K.S. 1995. *Environment Centered Analysis and Design of Coordination Mechanisms.* Ph.D. Thesis, Department of Computer Science, University of Massachusetts, Amherst.

[15] Durfee, E. H. 1996. "Planning in distributed artificial intelligence," G. M. P. O'Hare and N. Jennings, Eds, *Foundations of Distributed Artificial Intelligence*, Wiley InterScience.

[16] Ephrati, E., M. E. Pollack, and V. Sigalit, 1995. "Deriving Multi-agent Coordination through Filtering Strategies," *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, Montréal, Canada, Morgan Kaufmann Publishers Inc., pp. 679-685.

[17] Jennings, N. R. 1996. "Coordination Techniques for Distributed Artificial Intelligence," *Foundations of Distributed Artificial Intelligence*, G. M. P. O'Hare and N. R. Jennings (eds.), Wiley.

[18] Kamel, M. and H. Ghenniwa, 1994. "Coordination of Distributed Intelligent Systems," *Soft Computing: Fuzzy Logic, Neural Networks and Distributed Artificial Intelligence*, F. Aminzadeh and M. Jamshidi, ed., Prentice Hall, Englewood Cliffs, pp. 229-260.

[19] Kamel, M. and H. Ghenniwa, 1994. "A Quantitative Analysis of Coordination Complexity for DAI Systems," *Proceedings of the Canadian Workshop on Distributed Artificial Intelligence*, Banff, Alberta.

[20] Klein, K. 1995. "iDCSS: Integrating Workflow, Conflict and Rationale–Based Concurrent Engineering Coordination Technologies," *Concurrent Engineering: Research and Applications,* 3, pp.21–27.

[21] Klein, M. 1993. "Computer-supported Conflict in Concurrent Engineering: Introduction to Special Issue." *Concurrent Engineering: Research and Applications,* 2, pp.145–147.

[22] Klein, M. ed. 1993. *Concurrent Engineering: Research and Applications,* Introduction to the Special Issue on The Application of Multiagent Systems to Concurrent Engineering, 2.

[23] Kraus, S., J. Wilensky, and G. Zlotkin, 1995. "Multiagent Negotiation Under Time Constraints," *Artificial Intelligence,* (75), pp. 297-345.

[24] Lander, S. E. and D. Corkill, 1996. "Designing Integrated Engineering Environments: Blackboard-Based Integration of Design and Analysis Tools," *Concurrent Engineering: Research and Applications,* Special Issue on The Application of Multiagent Systems to Concurrent Engineering, 4, pp. 59-71.

[25] Malone, T.W. and K. Crowston, 1991. *Toward an Interdisciplinary Theory of coordination*, Technical Report, MIT, CCS TR# 120, SS WP# 3294-91-MSA.

[26] Malone, T.W. and K. Crowston, 1994. "The Interdisciplinary Study of Coordination," *ACM Computing Surveys*, 26(1), pp. 87-119.

[27] Peters III, J. F. and N. Sohl, 1996. "Coordination of Multiagent Systems with Fuzzy Clocks," *Concurrent Engineering: Research and Applications,* Special Issue on The Application of Multiagent Systems to Concurrent Engineering, 4, pp. 73-87.

[28] Prasad, B. 1996. *Concurrent Engineering Fundamentals: Integrated Product Development,* Prentice Hall, New Jersey.

[29] Prasad, B. 1996. *Concurrent Engineering Fundamentals: Integrated Product and Process Organization,* Prentice Hall, New Jersey.

[30] Prasad, et al. 1993. "Information Management for Concurrent Engineering: Research Issues," *CERA*, 1(1), pp. 3–21.

[31] Rasmussen, J. 1986. *Information Processing and Human-Machine Interaction: an approach to cognitive engineering*, Elsevier Science Publishing Company Inc..

[32] Rosenschein, J.S. and G. Zlotkin, 1994. *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*, The MIT Press, Cambridge, Massachussets.

[33] Sichman, J.S. and Y. Demazeau, 1995. "Exploiting Social Reasoning to Deal with Agency Level Inconsistency," *Proceedings of the First International Conference on Multi-Agent Systems*, SF, USA, AAAI Press/The MIT Press, Menlo Park, California, pp. 352-359.

[34] Shoham, Y., and M. Tennenholtz, 1992. "On the synthesis of useful laws for artificial agents societies," *Proc. of Tenth National Conference on Artificial Intelligence.* San Jose, California, pp 276-281.

[35] Sobolewski, M. 1996. "Multiagent Knowledge-Based Environment for Concurrent Engineering Applications," *Concurrent Engineering: Research and Applications,* Special Issue on The Application of Multiagent Systems to Concurrent Engineering, 4, pp. 89-97.

[36] von Martial, F. 1992. *Coordinating Plans of Autonomous Agents*, New York, Springer-Verlag, LNAI 610.

[37] Yeh, R. T. 1992. "Notes on Concurrent Engineering," *IEEE Trans. on Know. and Data Eng.* 4(5), pp. 407–414.

Sylvain Lizotte

Sylvain Lizotte received a B.Sc. in Computer Science and a M.Sc. in multiagent systems from Laval University, Quebec City, Canada. He is currently working as a consultant in telecommunication and real-time systems for Cogilog-CGL Technologies (Montréal, Canada). His research interests are in multiagent systems and in real-time systems.

Brahim Chaib-draa

Brahim Chaib-draa received in 1978 a diploma in computer engineering from École Supérieure d'Électricité (SUPELEC) de Paris (France), and the Ph.D. degree in computer science in 1990. He has been employed on many projects in Europe, Africa and in North America. Currently, he is Associate Professor at Laval University, Canada. His research interests include: multiagent systems, computational models of concurrent engineering, natural language for the interaction, formal systems for AI, real-time software. He is member of the ACM, IEEE and the AAAI and he is on the Editorial Boards of *Concurrent Engineering: Research and Applications (CERA)* and *The International Journal of Advanced Manufacturing Systems.*