A Logical Model for Coordination Rule Classes in Collaborative Sessions

José Martín Molina Espinosa, Jean Fanchon, Khalil Drira
LAAS-CNRS
7, Avenue du Colonel Roche
31077 Toulouse CEDEX 4 France
Email: jmmolina@laas.fr, fanchon@laas.fr, khalil@laas.fr

Abstract

This paper presents a coordination model allowing managers to define coordination policies in collaborative sessions. Our model allows different collaboration sites to be managed by maintaining consistency of the distributed management actions at the user and collaborative tool levels. The model manages multi-tools and multi-users collaboration sessions. It is based on a partial order representation of interdependencies in collaborative sessions and a logical specification of the required properties. We define three rule classes of coordination sufficient to specify the properties of interest. Our model is used to implement a software component for collaborative session management. We proceed by control of user-level management actions and by automated execution of tool-level management actions.

1. Introduction

Session management is a key issue in collaborative applications[3][6]. This topic constitutes a key element in the success of the collaboration, since it is in charge of defining the dynamical behaviour of collaborative sessions, including crucial functions such as enabling and controlling interaction among participants. In collaborative applications, the session management is a service, specifically developed for each new application. This service is very tied to the application's and the environment's functionalities. The opening and flexibility in this kind of services is very limited.

Session management is constituted of two stages: the session management scheduling and the session management execution[12]. Session management scheduling refers to actions necessary to configure sessions and to invite participants. The configuration consists in the determination of the list of participants, the list of applications to be employed, the network parameters, etc. The process of invitation makes possible to participants to be notified of the existence of a session. This process allows participants to accept or to reject session invitation. In the case of implicit meetings according to Edwards[6], this first stage is not executed, because actions made during this stage are replaced by other mechanisms that allow finding a session.

Session management execution refers to actions fulfilled during the session. These actions describe the behaviour of the participants and of the applications during the session. Actions in line are divided in actions related to session (to open, to close, etc.), to participant management (to join, to leave, etc.) and to application management (to start, to stop, etc.). The in line management actions set determines the behaviour of the session. For example, we can suppose that participant connection order determines the priority to access critical data objects.

Current session managers give very few possibilities of coordination rules definition. In current session management literature, we can find several works that center their approach in the definition of the relations among participants, applications and information, [6, 16, 14, 15]. Edwards [6] and Texier et al. [16] refer essentially to the definition of the access rights of participants and applications on data. Rodriguez et al. [14] describes the application architecture to determine data flows between producer-consumer components. Tata [15] defines coordination policies based on data access and synchronization contracts established between members of a virtual team. His model is centered in role managing and activity synchronization. It also supports the inference of access rules across a set of basic rules. We have centered our model on the use of participant and application events without depending on the data aspect. We consider that participant and applications atomic events are sufficient to manage distributed sessions. The coordination rules allow defining the behaviour of the session, such as the authorization or not of late comers participant connection. These rules allow to define the interdependencies between participants and applications. By example, by using coordination rules we can define the following collaborative session policies:

- Participants connection order: the allowed order is first the chairman, later the secretary and finally the rest of participants.
- Session opening: the session will be opened when 80% of participants are connected.

- Session connection: participants cannot be connected to a session that has been closed.
- Session disconnection: a participant can not be disconnected if he is in charge of one or more application servers.

This type of policies definition depends on session purposes and can change for every session instance. In this paper, we present a model for coordination dependencies between participants and applications in collaborative sessions. The aim is the supervision of participant and applications behaviour during a session.

The model includes two aspects, on the one hand the definition of collaborative sessions by means of labelled partial orders and on the other hand the use of first order formulas to specify properties corresponding to coordination rules for collaborative sessions.

1. The labelled partial orders turn out to be well adapted for the definition of collaborative sessions. Labeled partial orders (LPO) as models for the behaviours of concurrent systems [13] express exactly the causality relations between the events of a behaviour and define an order among the actions of participants and applications during a session. Communication diagrams or Message Sequence Charts [8] are examples of such LPO. The advantage of LPO w.r.t. the sequential (or interleaving) behaviours, is that some properties can only be verified on the partial order, as the immediate precedence (see enabling sentence in section 3) and cannot in general be verified on the interleavings.

2. The use of First Order Logic (FOL) formulae on these labelled partial orders to define the properties of interest ensures the embedding in a well defined theoretical framework. FOL is sufficient to express the three basic modalities which we define to specify collaborative sessions. These three modalities are expressed by: the precedence sentence, the inhibiting sentence and the enabling sentence. Coordination rules are result of the combination of these sentences with logical operators. FOL in partial orders is strongly related to linear temporal logic (LTL) [4][7], and should be sufficient in the future work to specify some properties at a particular point or configuration of a behaviour. Temporal logic on partial orders [1][2][4][7] are the cutting edge of Model Checking issues, and we can benefit of the ongoing work on the whole field.

Another advantage of the specification of a system by logic formulae on the behaviours is the induced independency w.r.t. a particular system model during this phase of the development. The only requirement for the model is to have a concurrent semantics (like Communication Diagrams or message sequence charts). Future work should target the verification of system models like Communicating Finite Automata or Petri Nets.

The session management tool set developed has allowed us to cover most of the aspects in session management, going from session preparation up to session execution and control, including tools and participants[11].

The model has a double interest: on the one hand, the definition of a formal model of dependencies for the execution and control of management actions, and on the other hand, the exploitation of the model in order to ensure the correct control and execution of management actions avoiding incoherent scenarios.

The rest of the paper is divided as follows: Section 2 presents our model of dependencies management in collaborative sessions. First we present the formalization of collaborative sessions by LPO, next we present the syntax and semantics of FOL. Section 3 presents the three sentences which form the base of the model. Section 4 presents an application: the basic coordination rules developed for a session manager developed within the European project DSE. In section 5, we develop the conclusions and the perspectives of our work.

2. Multi-tool and multi-user session coordination model

Our model is centered on the description of interdependencies controlling the whole actions made by actors during a collaborative session. We have identified four kinds of action interdependencies:

- Inter-application dependencies. This type of conflict relates to interdependencies among applications of different types. For example, consider a session involving three different applications: a videoconference, a sharing application tool and a floor control manager. An order is necessary to launch these applications. Indeed, the floor control manager must be launched at last because it manages the two others.
- Intra-application dependencies. This type of conflicts appears when there are interdependencies among the components which compose an application. For example, we can consider a sharing application tool which is composed by three types of components: one or more server components, a proxy component and one or more client components. Each type of component is defined and executed according to the participant's role during the session. There exist dependency relations among these three types of components and they cannot be started at the same time. The server components must be started first because they represent the basic suppliers of the application. Then the proxy component must be

started, it carries out connections towards the already started servers. Finally, it is the turn of client components to be started, those carry out connections towards the proxy.

- Inter-participant dependencies. The actions carried out by participants during a session can be also interdependent. By example, according to the role associated to each participant, they must arrive at the session in a specific order. The chairman must be connected before any participant, because he is responsible of accepting the participant connections.
- Participant-tool dependencies. There exist interdependencies among actions carried out by participants and by applications. For example, if a session does not accept latecomers, then the connection action of participants is related to open action made by the session manager tool.

Definition 2.0 (Session actors) We define $N = T \cup P$ as a finite set denoting collaborative actors, we employ the term *actor* as defined by [9]. These actors can be software tools (elements of T) or human participants (elements of P). We also define A as a finite set denoting management actions. These actions may concern either user or tool level. Such actions may be: join or leave a collaborative session for users or start/stop actions for a collaborative tool.

2.1 Collaborative sessions definition

Definition 2.1 (Collaborative Session) *A collaborative session that involves the set of actors N and the set of actions A is a labeled partial order* $p = (E_p, \leq_p, l_p)$ *where :*
E_p *is a finite set of events,*
$\leq_p$ *is a binary reflexive, antisymmetric and transitive relation over* E_p
$l_p : E_p \rightarrow N \times A$ *is a labeling function over the events set.*
An event e of E_p *is called an occurrence of the pair (actor, action)* $l_p(e)$.

The immediate precedence relation between events, denoted $x \rightarrow y$ *, is defined by:*
$x \rightarrow y$ *if and only if* $x <_p y \wedge \forall z(x \leq_p z \leq_p y \Rightarrow z=x \vee z=y)$

Where $<_p$ *denote the strict relation naturally deduced from* $\leq_p$.

2.2 Coordination rules definition

Definition 2.2 (Logic syntax and semantics) *We denote by FO(≤ , N × A) the first order formulas built on ≤ and the alphabet N × A, these formulas are defined by the grammar :*

$$\varphi \quad := \quad P_{(n,a)}(x) \mid x \leq y \mid \varphi \wedge \varphi \mid \neg\varphi \mid \exists x\, \varphi \mid \forall x\, \varphi$$

We shall note $\varphi(x_1,\ldots,x_n)$ *when*$(x_1,\ldots,x_n)$ *are free variables that may occur in a formula* $\varphi \in FO$. *Let* $p= (E_p, \leq_p, l_p)$ *be a collaborative session and* $e_1,\ldots,e_n \in E_p$ *we note* $(p,e_1,\ldots,e_n) \models \varphi(x_1,\ldots,x_n)$, *to mean that* φ *is true in p if* x_i *is given the value* e_i *for all of i = 1,…,n.*

The satisfiability relation $\models$ *is defined by:*

- $(p, e) \models P_{(n,a)}(x)$ *iff* $l_p(e)=(n,a)$
- $(p, e_1, e_2) \models x_1 \leq x_2$ *iff* $e_1 \leq_p e_2$
- $(p,e_1,\ldots,e_n, e_{n+1},\ldots,e_{n+p}) \models \varphi(x_1,\ldots,x_n) \wedge \psi(y_1,\ldots,y_p)$ *iff* $(p,e_1,\ldots,e_n) \models \varphi(x_1,\ldots,x_n$ *and* $(p,e_{n+1},\ldots,e_{n+p}) \models \psi(y_1,\ldots,y_p)$
- $(p ,e_1,\ldots,e_n) \models \neg\varphi (x_1,\ldots,x_n)$ *iff* $(p ,e_1,\ldots,e_n) \not\models \varphi (x_1,\ldots,x_n)$
- $(p,e_1,\ldots,e_n) \models \exists x\, \varphi(x,x_1,\ldots,x_n)$ *iff there exists* $e \in E_p$ *such that* $(p,e,e_1,\ldots,e_n) \models \varphi(x, x_1,\ldots,x_n)$
- $(p,e_1,\ldots,e_n) \models \forall x\, \varphi(x,x_1,\ldots,x_n)$ *iff for every* $e \in E_p$, $(p,e,e_1,\ldots,e_n) \models \varphi(x, x_1,\ldots,x_n)$

Particularly, if φ *is a sentence (*φ *does not contain free variables) then it describes a property of p and we note* $p \models \varphi$.

The derived operator on formulae $\Rightarrow$ *is defined as usual as :* $(\varphi \Rightarrow \psi) \equiv (\neg\varphi \vee \psi)$.

3. Basic coordination dependencies

We have identified and formalized three types of relations expressing dependencies involving tool-to-tool, user-to-user, user-to-tool and tool-to-user management actions. These relations are formalized by three coordination rule classes which are presented below.

Definition 3.1 (Precedence sentence) *For any pair of actions (n,a), (n',a') ∈ N×A, the precedence sentence, denoted by Pred((n,a),(n',a')), is defined as :*

$$Pred\big((n,a),(n',a')\big) \equiv$$
$$\forall x,\ P_{(n',a')}(x) \quad \Rightarrow \quad \big(\exists y,\ y < x \quad \wedge \quad P_{(n,a)}(y)\big)$$

We also define the following derived sentences :

$$Pred_n(a,a') \stackrel{def}{=} Pred\big((n,a),(n,a')\big)$$

$$Pred_a(n,n') \overset{def}{=} Pred((n,a),(n',a))$$

The precedence sentence is defined in order to specify the causal dependency among the occurrences of management actions. This sentence is interpreted as: actor n' cannot execute management action a' before management action a is executed by actor n.

Definition 3.2 (Inhibiting sentence) *For any pair of actions (n,a), (n',a') ∈ N×A, the inhibiting sentence, denoted by Inhib((n,a),(n',a')), is defined as :*

$$Inhib((n,a),(n',a')) \equiv$$
$$\forall x,\ P_{(n',a')}(x) \quad \Rightarrow \quad (\forall y,\ y < x \Rightarrow \neg P_{(n,a)}(y))$$

We define also the following derived sentences :

$$Inhib_n(a,a') \overset{def}{=} Inhib((n,a),(n,a'))$$
$$Inhib_a(n,n') \overset{def}{=} Inhib((n,a),(n',a))$$

The inhibiting sentence is defined to inhibit the execution of a given management action after a previous one is executed. This sentence is interpreted as: once management action a is executed by actor n then management action a' cannot be executed by actor n'.

Definition 3.3 (Enabling sentence) *For any pair of actions (n,a), (n',a') ∈ N×A, the enabling sentence, denoted by ImPred((n,a),(n',a')), is defined as:*

$$ImPred((n,a),(n',a')) \equiv$$
$$\forall x,\ P_{(n',a')}(x) \Rightarrow (\exists y,\ y \to x \quad \wedge \quad P_{(n,a)}(y))$$

We define also the following derived sentences :

$$ImPred_n(a,a') \overset{def}{=} ImPred((n,a),(n,a'))$$
$$ImPred_a(n,n') \overset{def}{=} ImPred((n,a),(n',a))$$

The enabling sentence is defined to ensure enabling a management action by another one. This sentence is interpreted as: management action a' must be executed by actor n' after the execution of management action a by actor n.

4. Coordination rules for DSE application

In this section, we present an application of our model to a real application. By using the three classes presented above we have modeled the coordination rules for the session manager developed within the frame of the Distributed System Engineering European project (DSE)[5][10].

4.1 Definition of actors

$$N_{basic} = P_{basic} \cup T_{basic}$$

The chairman is a special participant who is allowed to execute more management actions than other (general) participants. The chairman role can be played by different participants owning the chair attribute. But there exists in the participants set a unique participant who is the current chairman of the session, formally:

$$P_{basic} = \{chairman\} \cup \{p_i \mid 1 \le i \le n\}$$

The set of commonly used collaboration tools is composed of four tools:
smt: Session Management Tool,
gct: Group Conferencing Tool,
tms: Tool Management Service,
evs: Event Notification Service.
Formally:

$$T_{basic} = \{smt, gct, tms, evs\}$$

Where $A_{participant}$ and A_{tool} design respectively the participant-related and the tool-related management actions. Formally:

$$A_{participant} = \{join, leave, accept, message\} \text{ and}$$
$$A_{chairman} = \{grant\} \cup A_{participant}$$
$$A_{tool} = \{A_{smt} \cup A_{gct} \cup A_{tms} \cup A_{evs}\} \text{ with:}$$
$$A_{smt} = \{create, delete, open, close, invite\}$$
$$A_{gct} = \{enable, disable\}$$
$$A_{tms} = \{start, stop\}$$
$$A_{evs} = \{push, pull\}$$

The management of the basic coordination rules regarding session-related management actions has been implemented using GUI-level control (buttons and menus disabling). For execution of enabled actions we have implemented server-side remote invocations.

4.1 The core coordination rules

We identify in this section the basic coordination dependencies necessary for the coherence management in a multi-tools and multi-users collaboration session according to the set of roles and tools defined above. Three families of rules are defined: the session-state related coordination rules, the membership-related

coordination rules and the group and tool coordination rules.

4.1.1 Session state rules. The state sequencing dependencies for a collaborative session is presented in the rule δ_1. A collaborative session begins its life cycle after execution of creation event which sets session's state to initialized. After its creation, a session can pass to *announced* or to *deleted* states according to events carried out. Announced state means that the participants were invited to the session. While deleted state means the termination of the session. In addition, an announced session is either opened, and in this case it passes to *opened*, or it is cancelled, and in this case the session passes to *deleted*. The open event marks the beginning of collaborative work, this one must be enclosed by the close event. Collaborative work can be stopped while passing from *opened* to *deleted*. Once session has been *closed*, it is finished while passing to *deleted*.

$$\delta_1 = Pred_{smt}(create, invite) \wedge Pred_{smt}(invite, open) \wedge Pred_{smt}(open, close) \wedge \begin{pmatrix} Pred_{smt}(create, delete) \vee Pred_{smt}(invite, delete) \vee \\ Pred_{smt}(open, delete) \vee Pred_{smt}(close, delete) \end{pmatrix}$$

4.1.2 Membership rules. The following four coordination rules define the membership policies. These policies specify actions which can or cannot be made by participants according to the session's and participant's states. Figure 1. shows the transition-state machine defined conforming participants behaviour.

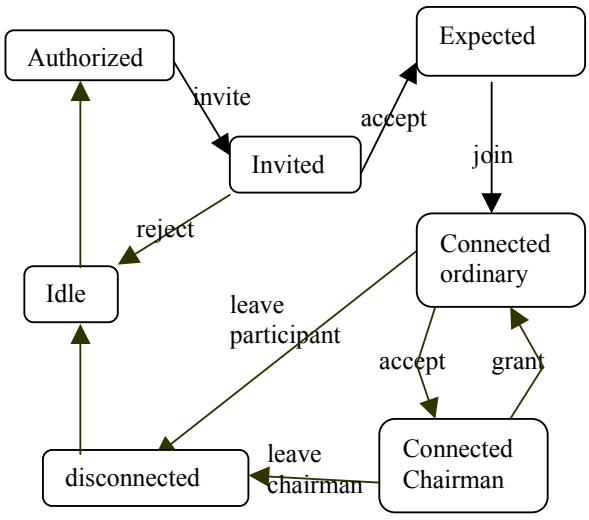

Figure 1. Participant's state machine

Participants join the session once they have been invited (rule δ_2):

$$\delta_2 = \bigwedge_{p \in P} Pred((smt, invite), (p, join))$$

Participants cannot join a deleted session. This rule avoids inconsistent request of connection to a session which does not exist any more (rule δ_3):

$$\delta_3 = \bigwedge_{p \in P} Inhib((smt, delete), (p, join))$$

Participants may not leave a session before joining (rule δ_4):

$$\delta_4 = \bigwedge_{p \in P} Pred_p(join, leave)$$

Only session members can communicate by sending and receiving messages (rule δ_5):

$$\delta_5 = \bigwedge_{p \in P} Pred_p(join, message)$$

4.1.3 Group and tools coordination rules. The following rules define the group behaviour and the applications coordination policies.

The chairman role can be granted only to the participant selected by the chairman (rule δ_6):

$$\delta_6 = \bigwedge_{p \in P} Pred((chairman, grant), (p, accept))$$

The new chairman must be connected(rule δ_7):

$$\delta_7 = \bigwedge_{p \in P_{basic}} Pred_p(join, accept)$$

The rule δ_8 defines a safety property, which consists in ensuring that an application can be stopped only if it has been really started. In consequence, the session manager must have in memory the list of the started applications.

$$\delta_8 = \bigwedge_{t \in T} Pred_t(start, stop)$$

Participants are disconnected automatically after deleting a session (rule δ_9):

$$\delta_9 = \bigwedge_{p \in P} ImPred((smt, delete), (p, leave))$$

Tools are stopped automatically after deleting a session (rule δ_{10}):

$$\delta_{10} = \bigwedge_{t \in T} ImPred((smt, delete), (t, stop))$$

Disconnected participants can not send messages (rule δ_{11}):

$$\delta_{11} = \bigwedge_{p \in P_{basic}} Inhib_p(leave, message)$$

5. Conclusion

We have presented a coordination model which defines the dependency relations for management actions during a collaborative session. We have identified and formalized three coordination rule classes: precedence, inhibition, and enabling. These three classes provide a powerful language to specify the consistency constraints during the execution of management actions between participants and tools. The proposed formal framework appears to be very expressive, ensures a rigorous modeling and facilitates further developments like temporal logic specifications and verification aspects. We have applied the model to define the basic coordination rules of the Distributed Systems Engineering project. We have implemented a session management package, which is compliant to the coordination rules we have presented in this paper. At this step of work, we do not manage multiple instances of the same tool. This constitutes our perspective for future work.

6. References

[1] Alur R., McMillan K., Peled D., "Deciding global partial-order properties". 25th International Colloquium on Automata, Languages, and Programming, LNCS 1443, 1998, pp. 41-52.

[2] Alur R., Peled D., Penczek W., "Model-Checking of Causality Properties", Proceedings of the 10th Annual IEEE Symposium on Logic in Computer Science (LICS'95), 1995.

[3] Constantini F. and Toinard C., "Collaborative Learning with the Distributed Building Site Metaphor", IEEE Multimedia, july-september 2001.

[4] Diekert V., Gastin P., "LTL is expressively complete for Mazurkiewicz traces", Actes de l'ICALP'00, Lecture Notes in Computer Science 1853, p. 211-222, 2000.

[5] Drira K., Martelli A., Villemur T., *Cooperative Environments for Distributed Systems Engineering*, Lecture Notes In computer Science 2236, Springer-Verlag, Berlin, 2001.

[6] Edwards, Keith W., "Session Management for Collaborative Applications", *Proceedings of the Conference on Computer-Supported Cooperative Work,* Chappel Hill, NC, USA, Oct 1994, pp. 32^3-330.

[7] Gastin P. and Mukund M., "An Elementary Expressively Complete Temporal Logic for Mazurkiewicz Traces", Proceedings of ICALP'02, Lecture Notes in Computer Science 2380, 2002, pp. 938-949.

[8] ITU-T, Message Sequence Charts (MSC 2000) ITU-T Recommendation Z120, 2000.

[9] Malone Thomas and Crowston Kevin, "The interdisciplinary Study of Coordination", ACM Computing Surveys, Vol. 26,No. 1, March 1994.

[10] Martelli A., "Distributed System Engineering*". Data Systems in Aerospace - DASIA - 2001 Symposium,* 28 May - 1 June 2001, Nice France.

[11] Molina Espinosa J.M., Drira K., Nabuco O., "A UML Model for Session Management in Collaborative Design for Space Activities". *In 8th European Concurrent Engineering Conference (ECEC 2001)*, Valencia, Spain, April 2001.

[12] Patterson, J.F., Hill, R. D., Rohall, S.L. and Meeks, W. S. "Rendevous : An Architecture for Synchronous Multi-user Applications". CSCW 90: Proceedings of the Conference on Computer-Supported Cooperative Work, Los Angeles, CA: ACM, 1990, pp. 317-328.

[13] Pratt.W. Modeling concurrency with partial orders. Int. J. Parallel Programming 15 (1987) 33-71.

[14] Rodriguez Peralta L.M., Villemur T., Drira K., Molina Espinosa J.M., "Managing dependencies in dynamic collaborations using coordination diagrams", 6th International Conference on Principles of DIstributed Systems (OPODIS'02), Reims (France), 11-13 Dec 2002, pp.29-42.

[15] Tata S., "Policies for Cooperative Virtual Teams", Proceedings 5th International Conference, COORDINATION 2002, York, UK, LNCS 2315, April 8-11, 2002 pp. 340-347.

[16] Texier G., Plouzeau N., "Automatic Management of Sessions in Shared Spaces", *Proceedings of the International on Parallel and Distributed Processing Techniques and Applications PDPTA'99;* CSREA Press, Las Vegas, Nevada, USA, June 28-July 1, 1999, pp. 67-73.

ACKNOWLEDGEMENTS

First author thanks Mexican Council of Science and Technology CONACyT for financial support through grant/loan 121900.