AN EVENT STRUCTURE BASED COORDINATION MODEL FOR COLLABORATIVE SESSIONS

Kamel Barkaoui[1], Chafia Bouanaka[2] and José Martín Molina Espinosa[3]
[1]*CEDRIC Research Laboratory, CNAM, Paris, France*
[2]*LIRE Laboratory, Mentouri University, Constantine, Algeria*
[3]*Tecnológico de Monterrey, Campus Ciudad de México, Mexico*

Keywords: Session Management, Collaboration, Event Structures, Computer Supported Collaborative Work.

Abstract: Distributed collaborative applications are characterized by supporting groups' collaborative activities. This kind of applications is branded by physically distributed user groups, who cooperate by interactions and are gathered in work sessions. The effective result of collaboration in a session is the production of simultaneous and concurrent actions. Interactions are fundamental actions of a collaborative session and require being coordinated (synchronized) to avoid inconsistencies. We propose in the present work an event structure based model for coordination in a collaborative session, making possible interactions between participants and applications in a consistent way. The proposed model describes interdependencies, in the form of coordination rules, between different actions of the collaborative session actors.

1 INTRODUCTION

Distributed collaborative applications are characterized by supporting activities of physically or virtually distributed groups which cooperate by interactions. These activities are performed by user groups and are gathered in work sessions. These sessions constitute the basic units of collaboration. The effective result of collaboration in a session is the production of concurrent actions carried out during definition and execution phases of the session. These actions require being coordinated in order to avoid inconsistencies.
Collaborative sessions represent a space in which various entities, such as participants, applications and data interact. Collaboration goal is to aid participants in the achievement of their tasks by giving them the possibility to exchange knowledge and information. These interactions imply that various actions of session actors tend to achieve a common goal. That is why actions might be complementary and assigned to actors with respect to their skills and /or functionalities. Interaction result, carried out during a collaborative session, strongly depends on the order in which they are executed. Indeed, interactions can be destructive if they are not coordinated.
Clearly, collaboration is the reachability of common goals of the underlying session. Reaching such goals depends on the respect of partial ordering between actions executed by all actors. Such order is depends on certain dependencies between actions. Using event concept, event occurrences obey to some coordination rules. Hence, we need formalism able to capture coordination between events. Ultimately, we require a formalism offering tools to express event occurrences with respect to causality, conflict or concurrency. Event structures (Winskel, 1987), (Winskel, 1992) are an adequate formalism to express mathematically these relations on sets of events. In this work, we present a model for collaborative sessions, based on the specification of interactions between actors during a session. The proposed model allows formal specification of different causal dependencies between events: precedence, inhibition, and release. This paper is organized as follows. We begin by a brief presentation of functional description of services implied in collaborative session management. In section 3, we recall essential aspects of event structures. In section 4, we introduce a new coordination model for collaborative session management, where basic coordination rules are well defined. In section 5, we present a mapping from event structures to Petri nets in order to exploit

Petri nets rich panel of tools for system verification. In section 6, we compare the proposed approach to other Models of collaborative sessions. We conclude by presenting directives for future work.

2 AN OVERVIEW OF CSCW

Computer Supported Cooperative Work (CSCW) refers to the study field concerned with design, adoption, and use of groupware (Ellis, 1994). Despite its name, this study field is not restricted to issues of cooperation or work but also examines competition, socialization and play. Computer Supported Cooperative Work is considered as a new research field involved in exploring a wide range of issues concerning cooperative work arrangements and support via information technology.

2.1 Actors of a Collaborative Session

The term session management refers to the process of starting, stopping, joining, leaving, and browsing collaborative situations across a network (Molina Espinosa, 2003b). Generally, we distinguish three types of users:

- A session participant: is a user taking part in a collaborative session and using applications assigned to session in progress.
- The session chair: is a participant who moreover controls session running (right to speak, tools access), invites participants, initializes, opens, and finishes the session.
- Session administrator: is responsible of defining and planning the session. He establishes participants list including session chair. These participants are selected according to their roles, their competencies, and their skills.

2.2 Functionalities of a Collaborative Session

We can identify two essential phases in a collaborative session: The first stage concerns session preparation. It consists of defining and updating participants list of the collaborative session and corresponds to Responsibility and Management Service (RMS). The second stage concerns session management and takes care of actions coordination during collaborative activities accomplishment. It is realised by Session Management Service (SMS).

2.2.1 Responsibility and User Management Service

RMS provides necessary functions to define participants profile, groups, and sessions (Molina-Espinosa, 2003a). It also provides information to *SMS* service and actually used applications.
RMS is composed of three main activities: Participants list definition, i.e., users intervening in the collaborative session. Session group's definition, each group represents a set of participants, and finally, establishment of active session list.

2.2.2 Session Management Service

SMS offers a set of mechanisms by which users can initialize, find, open, close, join, leave and finish a collaborative session. Basic functions of session management are:

- Session Initialization: corresponds to necessary objects creation by session chair. These objects are used to perform coordination actions during session evolution.
- Session Announcement: consists of inviting potential participants to a session. It also includes participants reply (Acceptance or Refusal).
- Session open and join: A session is opened by the session chair allowing participants to join it through their work stations.
- Session leave: allows participants to disconnect from a session.
- Session closure: allows session chair to terminate the collaborative work.
- Session termination: includes destruction of session objects, created during session initialization. This task is performed by session chair.

3 EVENT STRUCTURES

Generally, it is immaterial to analyze the precise places and times of event occurrences in a distributed computation (Winskel, 1992). Significant events and how their occurrence depends on the previous occurrence of other events are more important. Hence, distributed computations are considered as a set of event occurrences doted with a causal dependency relation between them. Defining a partial order between event occurrences is then reasonable. To model non-determinism and express how some event occurrences rule out the occurrence of others, a conflict relation between events is

defined. The set of events with causal dependencies and conflict relations are called event structures.
Event structures are abstract descriptions of a computation focusing on significant events of the computation and describing the possible ways that computation could follow.

Definition 1 (Winskel, 1987). A labelled event structure is a quadruple $ES = (E, \leq, L, \#, A)$ where:

- E is a finite set of events;
- $\leq \subseteq E \times E$ is a partial order relation called causal dependency relation which satisfies $\forall e \in E \, \{d \in E / d \leq e\}$ is finite;
- $\# \subseteq E \times E$ is an irreflexive and symmetric and relation, called conflict relation, which satisfies $\forall e, e', e'' \in E : e \leq e'$ *and* $e \# e'' \Rightarrow e' \# e''$;
- A is a set of actions;
- $L : E \rightarrow A$ is a labelling function on events, associating to each event *e* the corresponding action *L(e)*.

Concurrency or causal independency between events is a derived notion. Two events *e, e'* are concurrent and noted *e co e'* if and only if:

$$\neg((e \leq e') \vee (e' \leq e) \vee (e \# e')) \qquad (1)$$

A notion of computation state of an event structure can be defined (Winskel, 1992). Taking a process computation state as a set *X* of previously occurred events in the computation. We expect that:

- If an event has occurred this implies that all events on which it causally depends have occurred too.
- No two conflicting events can occur together in the same computation.

Definition 2 (Winskel, 1992). Let $ES = (E, \leq, L, \#, A)$ be an event structure. Defining its configurations $D(E, \leq, \#, L)$ consists of those subsets $X \subseteq E$ *which are*:

- Conflict free:

$$\forall e, e' \in X, \neg(e \# e') \qquad (2)$$

- Left Closure:

$$\forall e, e' \in E, (e' \leq e) \text{ and } (e \in X) \Rightarrow e' \in X \qquad (3)$$

These two conditions allow us to construct consistent configurations (executions).

4 A FORMAL MODEL FOR COLLABORATIVE SESSIONS

Collaboration aim is the attainability of common goals pursued by a session. Reaching such goals depends on respecting partial orders defined between actor's actions and induced by dependencies between actions. Using event concept, event occurrence might obey to some coordination rules. So, we need formalism able to capture coordination between events and offers necessary tools to express event occurrences with respect to causality, conflict or concurrency. To model the collaborative session by an event structure, we begin by identifying the set of inherent events and their relationships. During a session, actors perform some actions corresponding to some event occurrences. Hence, we define first the actors set; actions set, and define the concept of event thereafter.

4.1 Actors Set

We use T_{Basic} to denote the set of collaborative applications,

$$T_{Basic} = \{SMT, GCT, TMS, ENS\} \qquad (4)$$

- *SMT* is Session Management Tool. It allows session configuration, participants invitation, and session control;
- *GCT*, for Group Conferencing Tool, allows direct discussion between a group of participants;
- *TMS* is Tool Management Service;
- *ENS*, Event Notification Service, informs participants about actions made by other ones.

P_{Basic} denotes the list of session participants.
Actor set is defined by $N_{Basic} = P_{Basic} \cup T_{Basic}$. One participant in a session has the role of session chair.

$$P_{Basic} = \{SessionChair\} \cup \{pi \,/\, 1 \leq i \leq n\} \qquad (5)$$

4.2 Actions Set

Actions set A_{Basic} contains all actions performed during a session. A_p, A_{Chair}, A_T denote respectively actions performed by all participants, session chair and application tools. We have:

$$A_{Basic} = A_P \cup A_{Chair} \cup A_T \qquad (6)$$

where:

$$A_P = \{join, leave, accept, message\} \qquad (7)$$

A participant connects himself to the collaborative session by executing a join action. He leaves the session by a leave action and receives or sends a message to another participant via message action. Finally, a participant can accept the session Chair role by performing an *accept* action,

$$A_{Chair} = A_P \cup \{grant\} \tag{8}$$

grant action consists of passing the session Chair role to another participant,

$$A_T = A_{SMT} \cup A_{GCT} \cup A_{TMS} \cup A_{ENS} \tag{9}$$

A_T includes a variety of actions made available by services and applications used during a collaborative session. Actions ensured by Session Management Service are *create*, *delete*, *open, close* and *invite*.
A videoconference application, noted *GCT*, allows enabling or disabling a videoconference associated to a collaborative session by performing the *enable/disable* actions respectively. Management Groupware Service, *TMS*, allows starting and stopping groupware applications by carrying out *start* and *stop* actions. Event Notification Service, *ENS,* allows sending and receiving events through *push* and *pull* actions execution.

4.3 Event Definition

An event $e = (n, a)$, where (n, a) is a pair belonging to $N \times A$, is said to be occurred if action a is performed by actor n. We need the two following functions: *Act* and *Mgt*.

$Act : E \to N$, determines an event actor. It is defined by:

$$Act(e) = n \Leftrightarrow \exists a \in A \,/\, L(e) = (n, a) \tag{10}$$

$Mgt : E \to A$, determines an event occurrence associated action. It is defined as:

$$Mgt(e) = a \Leftrightarrow \exists n \in N \,/\, L(e) = (n, a) \tag{11}$$

4.4 Main Relationships

Dependencies between events can be categorized in three classes: precedence, inhibition and trigger relations. They are significant for coordination rules, between collaborative session actors, definition.

4.4.1 Precedence Relation

Precedence relation expresses the fact that executing some action *a'* (event *e'*) might be preceded by eventual execution (occurrence) of another action *a* (event *e*).

$$\Pr ed(e, e') \equiv \forall C, C' \in D : C \xrightarrow{e'} C' \Rightarrow e \in C \tag{12}$$

4.4.2 Inhibition Relation

Inhibition relation forbids the execution (occurrence) of an action *a'* (the associated event *e'*) if action *a* (event *e*) is already executed (occurred).

$$Inhib(e, e') \equiv \forall C, C' \in D : C \xrightarrow{e} C' \Rightarrow e' \notin C \tag{13}$$

4.4.3 Trigger Relation

To express immediate causality between events, we define trigger relation. That is, an event occurrence *e* follows immediately the occurrence of event *e'*. Formally, we write:

$$Trig(e, e') \equiv \forall C' \in D; C' \xrightarrow{e'} C'' \Rightarrow \exists C \in D \,/\, C \xrightarrow{e} C' \tag{14}$$

Finally, we define a session state concept. It can be viewed as a configuration in the corresponding event structure.

Definition 3. Let $(E, \leq, \#, L, A)$ be an event structure modelling the collaborative session. If *D* denotes the set of its finite configurations, then:

$$e' \leq e \Leftrightarrow \forall C \in D : e \in C \Rightarrow e' \in C \tag{15}$$

$$e \# e' \Leftrightarrow \forall C \in D : e \in C \Rightarrow e' \notin C \tag{16}$$

$$\begin{gathered} e \; co \; e' \Leftrightarrow \exists C, C' \in D : e \in C \; and \\ e' \in C \; and \; C \cup C' \in D \end{gathered} \tag{17}$$

Mapping from a session state to another is due to an action execution by an actor. It can be considered as a transition from configuration *C* to configuration *C'* when event *e* occurs. We write:

$$C \xrightarrow{e} C' \Leftrightarrow e \notin C \; and \; C' = C \cup \{e\} \tag{18}$$

4.5 Modelling Session Actors

The dynamic behaviour of a collaborative session requires expressing all constraints to respect during a session evolution. These constraints are called coordination rules and are expressed as follows:
Let $e_1, \ldots, e_n$ be events, *p* a participant, and *t* be an application; then we have:

4.5.1 Event Structure Associated to a Participant

Event structure associated to a participant *i* is defined by $ES_{Pi} = (E_P, \leq_P, \#_P, L_P)$ where:

- E_P is the set of events occurring due to actions performed by a participant, and equals to $\{e_{P1}, e_{P2}, e_{P3}, e_{P4}\}$. These events are labelled

respectively *(p, join), (p, leave), (p, accept), (p, message)*.
- Causal dependency expresses the fact that : Any participant can receive or send a message only if he was already connected, i.e. *(p, join) ≤ (p, message)*. Any participant can disconnect from a session only if he was already connected, i.e. *(p, join) ≤ (p, leave)*. Any participant can accept Session Chair only if he is a member of the group (connected to the session), i.e. *(p, join) ≤ (p, accept)*. So, we have:

$$\leq_P = \{(e_{P1}, e_{P4}), (e_{P1}, e_{P2}), (e_{P1}, e_{P3})\} \quad (19)$$

- Conflict relation expresses inconsistencies that might be forbidden : Any participant cannot receive or send messages as soon as he disconnects from a session, i.e. *(p, leave) # (p, message)*

$$\#_P = \{((p, leave), (p, message))\} \quad (20)$$

A Session Chair is a particular participant. He has the ability to grant this role to another participant. Thus, $ES_{chair} = (E_{chair}, \leq_{chair}, \#_{chair}, L_{chair})$ where:
- $E_{chair} = E_P \cup \{e_{chair}\} / L(e_{chair}) = (Chair, grant)$
- $\leq_{chair} = \leq_P \cup \{(e_{chair}, e_{p2})\} / L(e_{p2}) = (p, leave)$

A Session Chair cannot leave the session before granting his role to another participant. : $\#_{chair} = \#_P$

4.5.2 Event Structure Associated to Session Management Tool

SMT associated event structure is defined by $ES_{SMT} = (E_{SMT}, \leq_{SMT}, \#_{SMT}, L_{SMT})$ where: $\#_{SMT} = \varnothing$
- E_{SMT} is composed of five events e_{SMT1}, e_{SMT2}, e_{SMT3}, e_{SMT4}, e_{SMT5}, labelled respectively *(SMT, create), (SMT, invite), (SMT, open), (SMT, close), (SMT, delete)*.
- $\leq_{SMT} = \{(e_{SMT1}, e_{SMT2}), (e_{SMT2}, e_{SMT3}), (e_{SMT3}, e_{SMT4}), (e_{SMT4}, e_{SMT5})\}$

4.5.3 Event Structure Associated to an Application

Event structure associated to an application is defined by $ES_{TMS} = (E_{TMS}, \leq_{TMS}, \#_{TMS}, L_{TMS})$ where:
$E_{TMS} = \{e_{TMS1}, e_{TMS2}\}$, labelled *(TMS, start) and (TMS, stop)* respectively, $\leq_{TMS} = \{(e_{TMS1}, e_{TMS2})\}$ *and* $\#_{SMT} = \varnothing$

4.6 Coordination Rules

Collaborative session global behaviour corresponds to a parallel composition of individual behaviours (expressed by event structures defined bellow) of all actors coordinated by the following global rules to avoid inconsistencies: Coordination rules for participant admission and leave defines the coordinated behaviour of all participants of a session plus session management tool:
- Rule 1: Any participant admission to a session is allowed only if he was invited to that session: $\exists\ e_1 \in E_{SMT},\ e_2 \in E_P,\ L(e_1) = (SMT, invite) \wedge L(e_2) = (p, join) \Rightarrow Pred(e_1, e_2)$
- Rule 2: Any participant admission to a session is forbidden as soon as this session is closed : $\exists e_1 \in E_{SMT},\ e_2 \in E_P,\ L(e_1) = (SMT, close) \wedge L(e_2) = (p, join) \Rightarrow Inhib(e_1, e_2)$
- Rule 3 : Any participant is automatically disconnected from a session as soon as it is closed: $\exists\ e_1 \in E_{SMT},\ e_2 \in E_P,\ L(e_1) = (SMT, close) \wedge L(e_2) = (p, leave) \Rightarrow Trig(e_1, e_2)$

Coordination rule for session chair delegation is:
- Rule 4 : Any participant who accept the role of Session Chair becomes so only if he is invited to this role by the current chair: $\exists\ e_1 \in E_{chair},\ e_2 \in E_P,\ L(e_1) = (Chair, grant) \wedge L(e_2) = (p, accept) \Rightarrow Pred(e_1, e_2)$

Coordination rule for application integration to the session is:
- Rule 5 : Any application is stopped as soon as the session is deleted, i.e. $\exists\ e_1 \in E_{SMT},\ e_2 \in E_{TMS},\ L(e_1) = (SMT, close) \wedge L(e_2) = (p, stop) \Rightarrow Trig(e_1, e_2)$

5 SYSTEM VERIFICATION

Model verification is a crucial phase in system specification. It allows detecting eventual inconsistencies in the proposed model. Since Petri nets offer a variety of tools for analyzing system properties, we propose to exploit the mapping (Nielsen, 1981) between event structures and a sub-class of Petri nets, condition/event nets, to verify some properties of the event structure based model for collaborative sessions. The mapping is constructed at a categorical level through a coreflection composed of two functors. First funtor shows how to embed the category of event structures into that of Petri Nets. The other funtor abstracts away system implementation details, i.e. Petri Net places, to bring it back to an event structure. In this work, we are interested of the first functor, where an event structure is identifiable to an occurrence net ensuring causal and conflict relations defined in the event structure and obtained from execution of the condition/event net defined as follows:

Definition 4 (Nielsen, 1981). A condition/event net is a quadruplet (B,E,F,M_0) where:

- B is a set of non null conditions,
- E is a disjoint set of events,
- F is set of $(B\times E)\cup(E\times B)$ called causal dependency relation,
- M_0 is a non empty set of conditions, called initial marking.

The Petri net satisfies the following restrictions:

- $\forall e\in E, \exists b\in B / F_{b,e}>0$ *and* $\forall e\in E, \exists b\in B / F_{e,b}>0$; no isolated events
- $\forall b\in B$, $[M_{0b}\neq 0$ *or* $(\exists e\in E, F_{e,b}\neq 0)$ *or* $(\exists e\in E, F_{b,e}\neq 0)]$; no isolated conditions

Definition 5 (Nielsen, 1981). Let $E=(E,\leq,\#)$ be an event structure, the corresponding occurrence net is defined by $N(E)=(B,E,F,M)$ such that :

- $M=\{(\Phi,A) / A\subseteq E \text{ and } (\forall a,a'\in A,\ a(\#\cup 1_E)a')\}$
- $B=M\cup\{(e,A) / A\sqsubseteq E \text{ and } e\in E \text{ and } (\forall a,a'\in A, a(\#\cup 1_E)a') \text{and } (\forall a\in A, e<a)\}$
- $F=\{(e,(c,A)) / (e,A) \in B\}\cup\{((c,A),e) / (c,A)\in B \text{ and } e\in A\}$

6 RELATED WORK

Several models have been established for session management, such as CONCHA, GMS, Mediaspace and Intermezzo. CONCHA model (CONference system based on java and Corba event CHAnnels service) presented in (Orvalho, 1999) is a supervising authority of conferences based on CORBA events service. Services are implemented in Java and support reliable multicast communications for data transfer and information control. CONCHA includes two essential services: supervising conference authority and a multipoint communication service. In (Wilde, 1996), Group Management System for Distributed Multimedia Applications (GSM) model is presented. GSM is constitued of user agents and system agents. User agents are components being integrated in the group communication platform. System agents function is to manage distributed directories providing distributed databases to all user agents. In (Roussel, 1997), there is an inefficiency to collaborate in traditional media space systems because possibilities, to express coordination and actions, have been pointed out. Roussel proposed a new model, called Mediaspace, to handle collaboration. This model is rooted in a multi agents approach. Roussel characterized an agent by four fundamental properties: persistence, ability to/for communication, autonomy, and reactivity. In (Edward, 1994), a session management model based on sharing users and activities information is presented. Activities information include current tasks details, active tasks details (e.g., connected users), location of applications or tasks, and objects associated with these tasks. While CONCHA, GMS and Mediaspace models are centered on distributed entities (agents or components) integration, Edward's model is based on shared objects. All these models do not consider interdependencies management between participants and applications. Current session managers offer few possibilities for coordination rules definition. In the literature, there exist several works centered on the definition of relations among participants, applications and information (Rodriguez, 2002), (Tata, 2002). Rodriguez et al. (Rodriguez, 2002) describe application architecture to determine data flows between producer-consumer components. Tata (Tata, 2002) defines coordination policies based on data access and synchronization contracts established between members of a virtual team. This model is centered on role management and activity synchronization. It also supports inference of access rules across a set of basic data. We have centered our model on the use of participants and application events without depending on data aspect. Our approach consists in specifying a variety of dependency relationships during cooperative session execution. The proposed model is based on event structures, giving more clarity and formality to application specification. There are many similarities between the proposed model and Espinosa et al. (Molina-Espinosa, 2003b) one in defining dependency relationships. However, the major difference is the formal model used to specify cooperative sessions and coordination rules. While, Espinosa et al. used the Labeled Partial Orders (LPO) for collaborative session definition and First Order Formulas (FOL) to specify properties corresponding to coordination rules. We have used Event structures to specify both collaborative session and coordination rules. This allows us to discard ambiguity that appears in (Molina-Espinosa, 2003b) model.

7 CONCLUSIONS

In this paper, we have formally modelled collaborative sessions using event structures. The major benefit of such a model is the clarity of various aspects in session management. In fact, event structures give a theoretical foundation to our model, allowing formal verification of some properties. The work may be extended by proposing an automatic mapping to other models and formalisms such as Petri nets. Our current research is to specify coordination rules by an extension of event structures to domain event structures in order to manage multiple instances of the same tool. Mapping the specification expressed within event structures to operational framework is an other direction of future work.

REFERENCES

Edward W. K., 1994. "session Management for Collaborative Applications". Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW'94), Chapel Hill, NC.

Ellis C. A., Wainer J., 1994. "A Conceptual Model of Groupware". CSCW'94 Conference on Computer Supported Cooperative Work, pp. 79-88, ACM, Chapel-Hill, North Carolina, USA, .

Molina-Espinosa J. M., Fanchon J., Drira K., 2003a. "A logical Model for Coordination Rule Classes in Collaborative sessions", Rapport LAAS N03132, IEEE International Workshops on Enabling Technologies: infrastructure for collaborative entreprises, p.5, Linz, Australia).

Molina Espinosa J. M, 2003b. "Modèles et services pour la coordination des sessions coopératives multi applications: application à l'ingénierie systèmes distribués". Thèse de doctorat en Informatique et télécommunications, LAAS of CNRS, Toulouse.

Nielsen M., Plotkin G.D, Winskel G., 1981. "Petri Nets, Event structures and Domains". Part I of Theoretical Computer Science, Volume 13, N 1, pp. 85-108.

Orvalho J., Andrade T., Boavida F., 1999. "A conference Service Based on the CORBA Event Service". Proceedings of the 2nd Conference on Telecommunications, Telecommunications Institute de Portugal, Sesimbra, Portugal.

Rodriguez Perlta L.M., Villemur T., Drira K., Molina-Espinosa J.M., 2002. "Managing dependencies in dynamic collaborations using coordination diagrams". Rapport LAAS N 02527, 6th International Conference on Principles of Distributed systems, pp. 29-42, Reims, France.

Roussel N., 1997. "Au dela du Madiaspace: Un Modèle pour la collaboration médiatisée", In Actes des neuvieme Journee francophones sur l'Interaction Homme Machine (IHM'97), pp. 159-166, Futuroscope.

Tata S., 2002. "Policies for Cooperative Virtual Teams". Proceedings of the 5th International Conference, COORDINATION 2002, York, UK, LNCS 2315, pp. 340-347.

Wilde E., Freiburghaus P, Koller D., Platter B., 1996. "A Group and session Management System for Distributed Multimedia Applications, Multimedia Telecommunications and Applications". Third International COST 237 Workshop, Barcelona, Spain, LNCS 1185, pp. 1-22.

Winskel G., 1987. "Event Structures". In Advances in Petri nets, LNCS N. 255, pp. 325-392, Springer-Verlag.

Winskel G, Nielsen M., 1992. "Models for concurrency". The Handbook of Logic in Computer Science, Technical Report DAIMI PB-429, Computer Science Department, Aarhus University.