

# A UML MODEL FOR SESSION MANAGEMENT IN COLLABORATIVE DESIGN FOR SPACE ACTIVITIES\*

José Martín Molina Espinosa<sup>1</sup>

Olga Nabuco<sup>1,2</sup>

Khalil Drira<sup>1</sup>

<sup>1</sup>LAAS-CNRS

7, avenue du Colonel Roche  
31077 Toulouse Cedex 4 France

<sup>2</sup> Centro de Pesquisas Renato Archer - CenPRA

Rodovia SP-65, Km 143.6 13089-500 Campinas – SP Brazil

E-mail: jmmolina@laas.fr, ofnabuco@laas.fr, khalil@laas.fr

## KEYWORDS

Computer Supported Cooperative Work, Session Management, UML Language.

## ABSTRACT

This paper describes the Session management protocol designed in conformance to the design activity requirements as specified for Space Scenario activities formalized within the European project Distributed Systems Engineering (DSE 2000). The Session Management is deployed using Unified Modeling Language allowing its implementation by object oriented languages, also offering a comprehensive model to be followed by team developers and user's representative.

## INTRODUCTION

Enabling collaboration between people since early phases of large projects is one of the challenges faced by the teams in charge of developing collaborative environments. The goal is to reduce time on design and verification phases taking advantage from the core competence of each partner independently of the Time Zone and the software application.

The team involved in the development faces the same challenge. Users and developers shall be involved since the beginning. To achieve this requirement, the partners shall have a transparent communication channel, with people's well-defined roles and responsibilities [Moenaert 2000]. In DSE project, the collaborative environment's requirements were elaborated by the user's representative, providing directions to the development's team. The scenario, presented by the user, was the revision phase of the Automated Transfer Vehicle (ATV) project. During this phase, specialists from diverse domains shall work together to analyze the documents (of different sources, like CAD, text, specifications, etc) delivered by the ATV designers in order to search design discrepancies.

On the analyses of these requirements, DSE's partners have concluded to build work group facilities focused on the fact that end users want a single environment able to deal with

various cooperation tools. The facility represented by the Session Management, enables the formation of groups in distributed meetings. The term session management refers to the process of starting, closing, joining, leaving, and browsing collaborative situations across the network (Patterson et al. 1990). The Session model provides a unique interface that supports the user in scheduling the meetings, choosing the tools (application programs) that will be used, allowing the coordination of the activities and promoting a communication channel where information can be exchanged.

The UML (Booch et al. 1999) methodology has been chosen as a common language between the various developers involved in the DSE project. The resulting model allows a comprehensive view to the user representative, a fast prototyping to the developers and eases the integration of the different modules.

The UML Use Case View and the Logical View have given the bases for the DSE Architecture. The first one, Use Case View, offers a general perspective over the entire system. Each layer of the system is defined showing its main capabilities, the interaction between the layers, the services needed by each layer regarding the others. During the design of this view DSE developers' team could interact with each other defining 1) the tasks inside the modules, 2) the responsibilities in developing each module. Also, the user's representative could follow if the requirements are covered by the modules. This first interaction is important in order to define the communication channels, meaning, what task will be covered by what partner and who is responsible of what. The use of UML facilitates the exchange of information because, at that point, the actors involved in the system as well as the packages are determined.

The Logical View of the system defines 1) the architecture of the modules, 2) representative classes of each module, 3) the resources needed by each module, 4) the behavior of each Use Case and the steps to complete the tasks, represented by sequence diagrams.

The Session model is based on a multi-tiers architecture where entities can be resident in distant heterogeneous

---

\* This work is part of the European project DSE funded by the European Community under grant number IST-1999-10302 involving Alenia Spazio, EADS, SILOGIC, SIA, D3 Group, IABG, LAAS-CNRS and LIP6.

platforms. The object-oriented model, we have developed, allows different classes and relationships to be added in order to customize the session management protocol to diverse user needs.

The presented model defines the actors involved in the collaborative design activities in space domain, the life cycle of the session management module and the components identified for its implementation.

This implementation will be validated by the Preliminary Design and Review (PDR) processes for the ATV. ATV is used to supply the International Space Station with water, food, and equipment.

The remaining of this paper is presented like that: section 2 defines the Session Use Case requirements. Section 3 discusses the Session behavior and Session 4 presents the Session Logical View model.

## SESSION USE CASE VIEW

The scenario PDR is one of the program reviews that occur during the ATV (and others) project life cycle. During the program review phase, different skilled specialists, playing different roles, work together. The main goals of PDR process are to bring difficulties and potential risks to light in order to reduce them and to put documents and products in a reference database corresponding to the end of the phase. Three different levels of groups compose the PDR Review Groups: 1) the Review Board – they play the role of customer representatives deciding whether implementing or rejecting recommendations; 2) Project Team Interface – its members act as communication channels between designers and reviewers; 3) Review Committee – composed by different groups where experts, external to the project, recommend actions and generate Review Item Discrepancy (RID) documents. All those people shall work together in order to perform their tasks on time: 1) to prepare the Review (to compose PDR review groups, to assign review tasks of each one in these groups, to schedule the review meeting, to collect documents to review); 2) to execute the Review (supervision of review meetings, organization and management of RIDs) and 3) process the Review results (evaluation of review results by the review board). Each task maybe composed of several meetings. Following this scenario, a first design approach of the desired environment was elaborated.

### Session Definition

A Session involves a set of *actors* acting on a set of distributed *session objects*. *Actors* belong to *Review Groups* and may have one of the following three roles: *chairman*, *secretary*, or *participant*. *Session objects* represent recommendation documents, RID documents, or baseline objects such as: CAD files and simulation results.

Both actors and session objects and their relationship are defined in terms of classes. Actor is a parent class used to define participants profile including the role they will ensure in the collaborative design activity. The chairman role includes responsibility for session scheduling related actions including: selecting and inviting participants, defining

application-domain tools, and session management related action. This includes: opening, starting, stopping and deleting sessions. The same strategy in handle people-to-people collaboration is defined by (Peña-Mora et al. 2000). Participant role includes responsibility for session joining and leaving. These actions are provided by the session management service, the main component of our model. Other coordination functions such as session state saving and resuming are delegated to external modules implemented within the other layers of the DSE environment.

### Use Case View

Figure 1 depicts the UML use case view for the session model. This diagram shows the actors that will interact with session modules. We distinguish two types of actor: session chairmen and participants. Session chairman is responsible of initializing and finalizing sessions; these actions are related to objects instantiation and deletion. Also, a chairman manages a session: open/close sessions, accept/reject participants. Participants accept/refuse an invitation and join/leave a session.

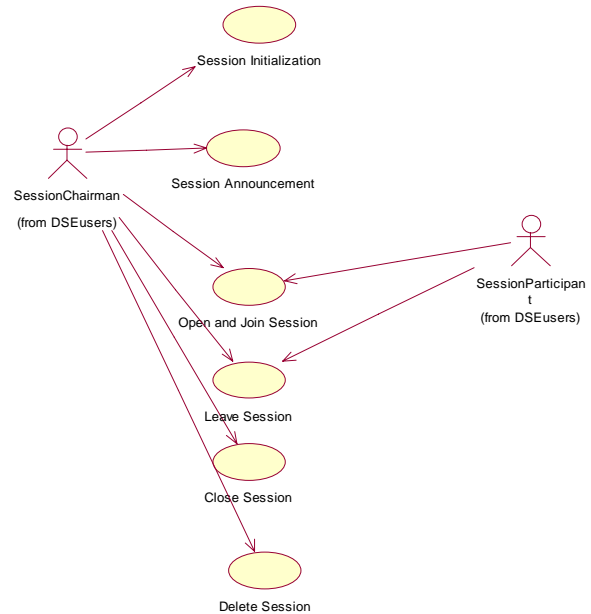


Figure 1: Session Model Use Case Diagram

## SESSION LIFE CYCLE

To model dynamic behavior of Sessions Management Service (SMS) we employed UML State Transitions Diagrams. The Figure 2 depicts the typical behavior of a SMS object. A SMS is created to handle a new session as the consequence of the initialize action taken by the session chairman. This brings the session component from defined state to initialized state. Once crated SMS is used to announce the session to the involved participants. In our implementation this announcement is made via an email to each involved participant. The mail contains the session name, password, date and time start, duration, theme, and the applet's URL for answering the invitation.

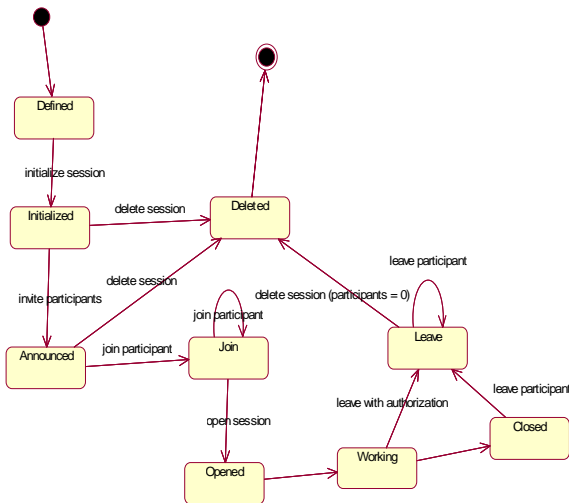


Figure 2: Session Management Service State Transition Diagram

When receiving this announcement each participant (via WEB page) can decide either to accept or to refuse the call to join the session. When all participants' responses have been collected by the chairman, the latter may decide either to open or delete the session. This depends on the number of

dedicated to post-collaboration data processing. At the end of this activity, the object is deleted.

### SESSION LOGICAL VIEW

We defined six collaboration diagrams that represent the behavior of each module as defined in the use case view (Figure 1). Each collaboration/sequence diagram comports the tasks to be executed by actors and modules to accomplish module functionality. Collaboration diagrams also represent resources allowed and interactions with others DSE sub-systems. Figure 3 shows the sequence diagram that depicts the module Session initialization. It is composed of three modules (Responsibility Manager, Session Management Service Factory and Session Management Service) and one actor (Session Chairman). To initialize a session, the chairman have to perform the following tasks: 1) selection of a session template in order to set main characteristics like periodicity; 2) creation of attended participants list; 3) selection of tools to be used during the session (conference tools, applications sharing tools, etc); 4) invocation of session initialization request to the Session Management Service Factory. The factory launches the creation of a Session Management Service.

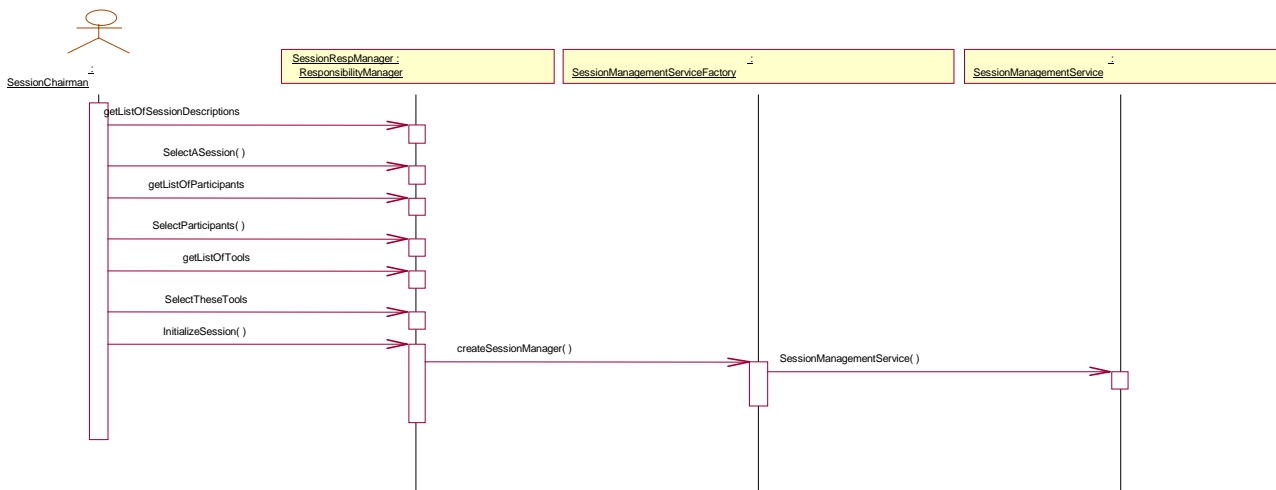


Figure 3: Initialization Session Sequence Diagram

positive responses. When the session is opened, a message is sent to all involved participants. Collaboration activity is then started. At this point, all the involved participants launch their cooperation tools such as videoconference applications, sharing applications, simulation programs. During this state the participants work together in order to achieve a common goal i.e. the preparation of the PDR review. During this phase participants may leave the session after the chairman approval. A list of actual participants helps chairman to decide whether a participant can leave the session.

At the end of the activity, the chairman declares the session closed and informs all participants by sending a specific message. This means the end of the coordination phase. After this announcement, participants may leave the session without asking the chairman's approval. The SMS is then

### Session Classes

The Class Diagram of the Session model is depicted by Figure 4. This figure shows the five main UML classes that represent the core of the Session model. These classes can be extended in order to adapt them to specific needs. The model was defined using a client-server approach. Current implementation was made using JAVA-RMI protocol. In the server side we defined the following classes:

- SessionManagementFactory (SMF): This class is used as a factory to define, to instantiate and to manage SessionManagement- Services objects. There is only one object of this class in the whole system. This class receives client's requests to create, to delete and to get references of SMS objects. Extension of this class will make possible session post processing, statistics, etc.

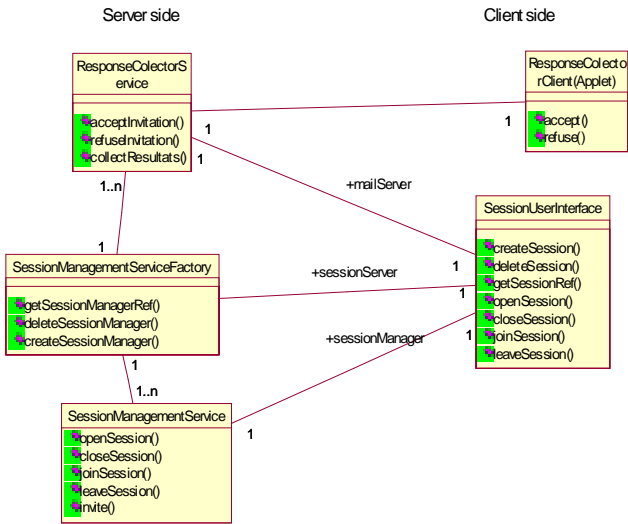


Figure 4: Session Model Class Diagram

- SessionManagementService (SMS): this class defines the operations provided to the collaborating actors by the support environment. This class plays the main role in the Session model. SMS class allows clients to open, to close, to join, to leave, and to invite to the current session. It also provides the collection of participants' responses as well as the creation of session confirmation WEB page.
- ResponseCollectorService (RCS): This server collects participants' responses in order to forward to the session chairman the list of participants that have accepted or refused the participation to the announced session. Participants' responses are collected via an applet defined in session confirmation WEB page.

In client side we have defined two main classes. The former is a graphical user interface that acts like a client for the server classes listed above. The latter is a class implemented as an applet to collect responses:

- SessionUserInterface (SUI): This class is a GUI that allows clients to access SMF, SMS and RCS. This class defines the allowed actions for each participant depending on his profile and collaboration state. Figure 5 shows a snapshot of this class. SUI class was integrated into the DSE Responsibility & Session Management. This DSE component manages a database to stock participants, roles, groups, organizations, and sessions information. DSE Responsibility & Session Management component furnishes session information to Session model.
- ResponseCollectorClient (RCC): This class defines the remote access to RCS by remote participants. This class is employed as an applet within session confirmation WEB page that allow actors to accept/refuse a session invitation.

## CONCLUSIONS

The Session model presented in this paper has been defined using UML and implemented in JAVA and distributed using the JAVA-RMI protocol. The current implementation includes 12 Java classes corresponding to about 2500 lines. The validation scenario will involve three partners located in Toulouse, Paris and Les Mureaux connected using an ATM high-speed network and two partners located at Turin, and Berlin connected using ISDN lines.

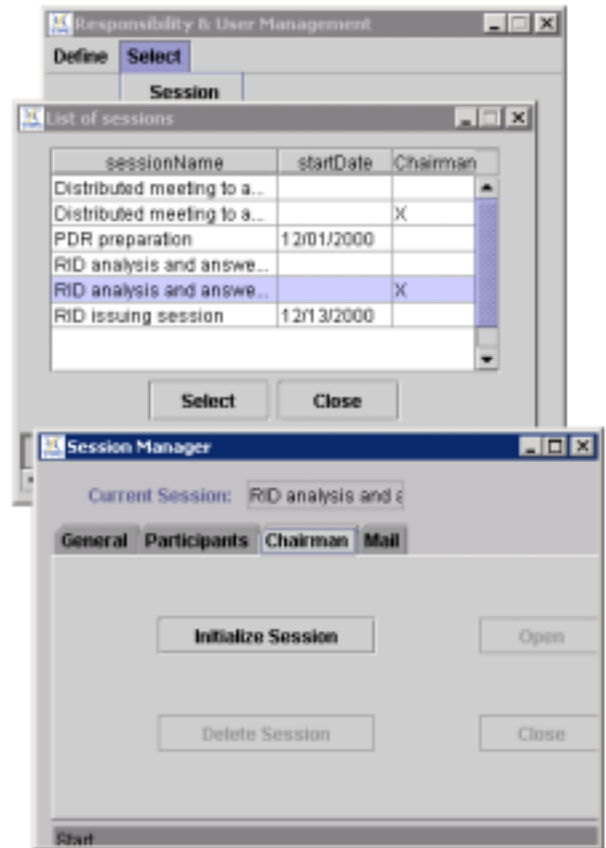


Figure 5: Snapshot of the Session Management Service Graphical User Interface

The requirements were provided by end user's representative, involved, since the early beginning of the project, in its conception and analyses. From conception to implementation of the modules, the user of UML has proved its integrability capacity, allowing mutual understanding between users and developers.

## REFERENCES

- DSE Project <http://cec.to.alespazio.it/DSE/>
- Booch G., Rumbaugh J., Jacobson I. 1999. "The Unified Modeling Language User Guide". Addison Wesley.
- Moenaert, R., Caeldries, F., Lievens, A., Wauters, E. 2000. "Communication flows in international product innovation teams." *Journal of Product Innovation Management* 17, 360-377.
- Patterson, J.F., Hill, R.D., Rohall, S.L., and Meeks, W.S. 1990. "Rendezvous: An Architecture for Synchronous Multi-user Applications." *CSCW 90: Proceedings of the Conference on Computer-Supported Cooperative Work*, Los Angeles, CA: ACM, 1990, 317-328.

Peña-Mora, F., Hussein, K., Vadhavkar, S., and Benjamin, K. 2000. "CAIRO: a concurrent engineering meeting environment for virtual design teams." *Artificial Intelligence in Engineering* 14 Elsevier Science Ltd., 2000, 203-219.

## ACKNOWLEDGEMENTS

First author thanks Consejo Nacional de Ciencia y Tecnología de México (CONACyT) for financial support through grant 121900.

Second author thanks Brazilian Ministry of Education (Fundação CAPES) for its support through a fellowship grant.

We thank our DSE partners and more particularly EADS for early discussion on these models. We also thank Ms. Beatrice Cambou for the use of Responsibility Manager graphical user interface.

## BIOGRAPHY

**J. Martin Molina E.** was born at Zacapu, Mexico. He is a Ph.D. student at LAAS-CNRS. He obtained his Master degree in computer science from INPT, the National Polytechnic Institute of Toulouse, in 1999. He received the Engineer degree in Computer Science from ITM, Institute Technologic of Morelia in 1994. He has worked for four years at the Mexican Electrical Research Institute on Simulation Department. His interest research areas are CSCW, distributed environments and cooperation.

**Olga Nabuco** was born at Rio de Janeiro, Brazil. She is in a Ph.D. sandwich program at LAAS-CNRS. She obtained her M.S. degree in Mechanical Engineer from Unicamp University, in 1996. She graduated in electrical engineer at Federal University of St Catarina in 1980. She has been working at Information Technologies National Institute as full-time researcher since then. Her interest research areas are Knowledge Sharing, Agent Technologies and CSCW.

**Khalil DRIRA** received the Engineer degree in Computer Science from ENSEEIHT/INPT, the National Polytechnic Institute of Toulouse, in June 1988, the M.S. degree (DEA) in Computer Science (Concurrent Systems orientation) from INPT in September 1988, and the Ph.D. degree in Computer Science from UPS, University Paul Sabatier Toulouse, in October 1992. He is since 1992 (tenured at 1993), Chargé de Recherche CNRS, a full-time research position at the National Center for Scientific Research of France. He is or has been involved in several national and international R&D projects in the field of distributed and concurrent communicating systems. His research interests include formal design and validation of concurrent and distributed component-oriented systems.