IMPLEMENTATION OF GRID CONFERENCES USING JADE TECHNOLOGY

Ajanta De Sarkar and Soumilla Sen

Department of Computer Science & Engineering, Birla Institute of Technology, Mesra, Kolkata Campus, Kolkata 700 107
adsarkar@bitmesra.ac.in, soumillasen@gmail.com

ABSTRACT

In grid environment, resources and users from different administrative domains are integrated and coordinated each other. Grid uses open standard, general-purpose protocols and interfaces to provide nontrivial quality of services. Videoconferencing is communication between two or more people from geographically different locations by simultaneous two-way video and audio transmissions. This is a type of 'visual collaboration' with audio-video communication. It differs from telephone calls through video display. Audio and Video communications is achieved in form of conferencing through Internet. This paper presents details of implementation techniques of Grid conferences using Jade technology. The detailed design of this Video Conferencing Solution (VCS) is based on Gaia methodology followed by UML techniques for easy understanding of the users. This conferencing solution enables the users to explore the power of collaborative conferencing in grid environment.

KEYWORDS

Gaia methodology, UML technology, Jade, Grid, video conferencing

1. INTRODUCTION

A computational grid is designed to deliver larger computational resources to the users within affordable cost. Grids are used to solve large-scale computational and data-intensive problems by creating virtual organizations, sharing, accessing and combining different sets of resources and specialized devices in secured way. Occasionally, audio-video based collaborative conferencing might necessary in dynamic and diverse resource pool of grid environment. Thus the execution of video conferencing in such a dynamic and heterogeneous environment is a challenging problem. The proposed video conferencing solution (VCS) is based on a Multi-Agent System (MAS).

VCS is a collection of autonomous interacting software agents can offer services among distributed computational resources through deployment of these interconnected agents onto the resources. These resources are available and allocated in grid environment. Major task of *VCS* is to create super administrator, group administrator and user. *VCS* users complete registration for joining a meeting as host or participant as well as leaving meeting. Super administrator is solely responsible for creating groups, license for the groups and administrator for group of users. Group administrator manages meeting, users of the group. Details of these functionalities of all these roles are discussed in [12].

In this research work, Gaia methodology has been used to carry out the detailed design of the individual agent structure and the multi-agent development process in the *VCS*. Therefore different types of analysis and design phases of *VCS* have been presented in [12] with *environment model*, *role model*, *interaction model* and *service model* respectively. Organization of this paper is as follows. Related work is presented in Section 0. Section 3 presents detailed design of the video conferencing solution using Gaia methodology. In contrast, agent interactions of VCS are presented in Section 4 using UML technology. Part of VCS is implemented and implementation details are discussed in Section 5. Section 6 concludes the paper with direction of future work.

2. Related Work

In recent times there has been a surge of interest in network based video conferencing system. Research and development in this area has expanded significantly in the past few years. The various popular available solutions may be enumerated as follows: Skype [4], PlaceCam [3], Vennfer [5], Woovoo [6], GoToMeeting [2] and Dimdim [1]. These products are Internet based and supports two way live audio video communications. The number of participants supported, codec, protocol, bandwidth requirement varies from product to product. Some products also provides extra feature like white board, test massaging, audio-video archiving and playback.

In [13], heterogeneous multimedia clients can join in the same real-time sessions. This Global Multimedia Collaboration System provides support for a variety of protocols and applications, including H.323 clients, SIP clients and Access Grid rooms. It facilitates audio and video communications among participating clients in a real-time conference. A roadmap for the future of videoconferencing within e-science is presented in [8]. In this research, the concept of Access Grid Studio-based Videoconferencing is studied. The proposed *VCS* supports Unicast and Multicast both environments with audio / video standard (AAC / H.264) respectively. *VCS* uses HTTP (Hyper Text Transfer Protocol) for communication with meeting server and RTP (Real Time Transfer Protocol) for audio-video streaming. The total minimum bandwidth requirement for *VCS* is around 150Kb/s while an optimum bandwidth requirement is around 300Kb/s. Novelty of this research is that it is based on multi-agent system (MAS) technology and proposed to be implemented in grid environment.

Comparing all the agent-oriented methodologies is often difficult, because they might address different aspects in different scenarios. In this research, Gaia methodology is specially chosen to design the system in details. Main focus of Gaia is on individual and autonomous agent structure and interaction of these agents in organized society to achieve a common specified goal. Gaia [16] does not explicitly deal with the activities of requirement gathering phase and implementation issues. As requirement gathering is already done in this research, analysis and design is discussed elaborately in this paper. Part of the implementation of this *VCS* is done with the FIPA-compliant agent platform, Jade framework [11] in Grid environment.

3. Design of Video Conferncing Solution

Video Conferencing Solution is designed with a group of inter acting multiple autonomous agents, which performs several tasks to provide the live conference among group of people from different locations. A software engineering approach, namely Gaia methodology has been considered for detailing the design of the *VCS*. The steps for modeling the system using this methodology are described in this section. It composed of three different phases, namely: *analysis phase*, *architectural design phase* and *detailed design phase*. Analysis phase briefs about corresponding roles of each sub organization. In architectural design phase, rules of liveness and safety described. It also finalizes role model and interaction model. At last, agent model and service model are developed in detailed design phase.

Analysis Phase

According to Gaia, the first step in the analysis phase of *VCS* is to determine whether multiple organizations have to coexist in the system and perform autonomously. Sub-organizations need to interact loosely with other segments of the system towards achievement of a given sub-goal.

The two major sub organizations of the system can be defined in analysis phase and the goal of each sub-organization is shown in Table 1.

Table 1: Sub-organizations in the Video Conferencing Solution

Name	**Description**
Meeting Provider	This sub-organization provides Group, License, User, Multicast IP, Meeting and Client Installer software
Meeting Executor	This sub-organization executes the meeting by enabling audio video interchange and desktop sharing

The major output of the analysis phase is the Environment Model. In this model, the corresponding roles in specific sub organization are modeled and the roles are as follows:

i. SuperAdministrator (SA), GroupAdministrator (GA) and User in *Meeting Provider* sub organization
ii. Host, ActiveParticipant (AP), PassiveParticipant (PP) and Transponder (TP) in *Meeting Executor* sub organization

However, the organizational rules of the above mentioned sub-organizations are also defined along with liveness and safety rules in [12].

Architectural Design Phase

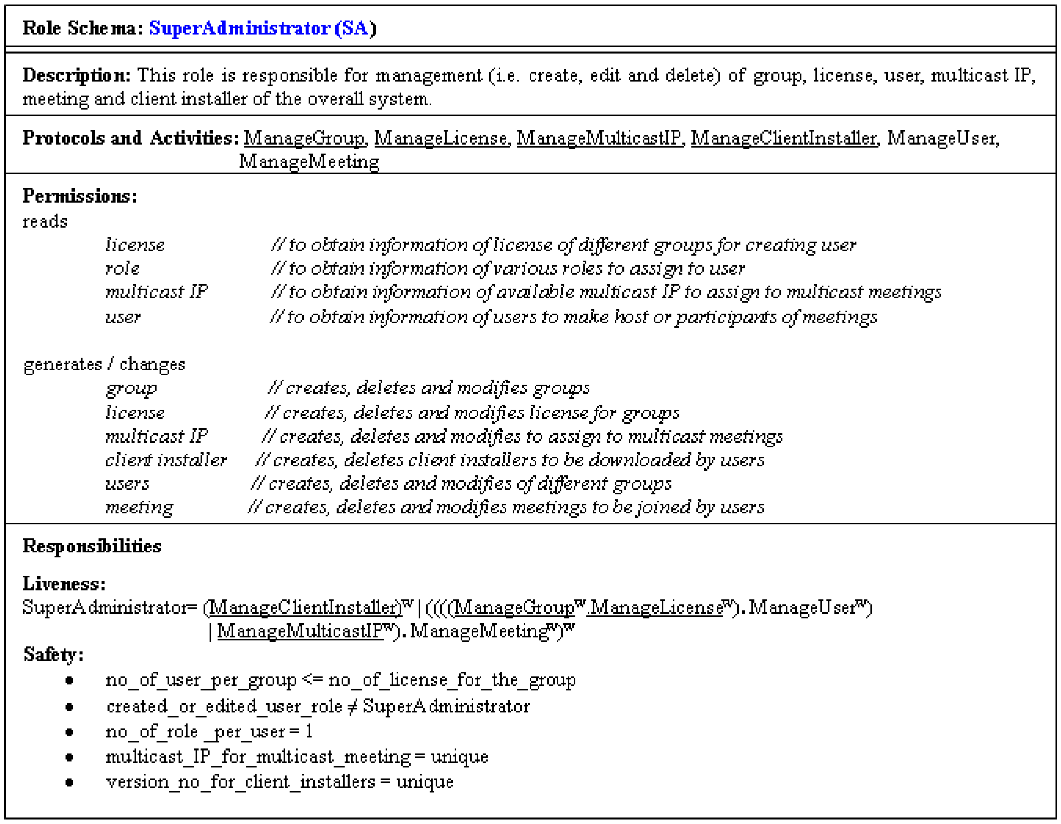
On the basis of the analysis, this section describes the architectural design of the *VCS* that completes and refines the preliminary models and makes actual decisions about the organizational structure and models the *VCS* based on the specifications produced.

As it is already mentioned that *Meeting Provider* sub organization is responsible for the overall provisioning which includes management of Group, License, User, Multicast IP, Meeting and Client Installer software. There are three types of roles in the system SuperAdministrator, GroupAdministrator and User. There is only one SuperAdministrator in the whole system. SuperAdministrator creates a new group for the customer or user and licenses are created whenever a new customer purchases license. SuperAdministrator will also create one User with GroupAdministrator role for the group. SuperAdministrator may create number of Users less than equal to the licenses being purchased for the group. Alternatively GroupAdministrator also may create the other Users.

Meetings may be of three types Multicast, Unicast, Multicast-Unicast. SuperAdministrator also creates a number of unique Multicast IP for the group which will be required to create multicast and unicast-multicast meetings. SuperAdministrator, GroupAdministrator and User any one can create meeting by selecting the host, participant and Multicast IP (for multicast and unicast-multicast meeting types). User may edit and delete meetings created by him/her. GroupAdministrator may edit and delete users and meetings of his/her group. SuperAdministrator also manages the Client Installer software as well as edit and delete Group, License, User, Multicast IP, and Meeting.

Meeting Executor sub organization is responsible for running the meeting participated by host and other participants. Running meeting includes organizing and hosting meeting by enabling audio video interchange and desktop sharing. Host transmits his/her audio video to all other participant of the meeting. It also can share his/her desktop to other participants to give a presentation. Participant can raise their hand to become "active". Host may select any of the participants as "active". ActiveParticipant (AP) also transmits his/ her audio video to host as well as other participants. Transmitted audio video may reach directly to the recipient or via n number of hops in-between depending upon the distance between the sender and receiver.

Next on the basis of the analysis and organizational structure, the agent role model of the *VCS* is described. The role schemas are presented using the templates from [16, 15]. At first, roles of the *Meeting Provider* sub organization are discussed. SuperAdministrator role has the ultimate authority of the system. SuperAdministrator performs these activities Group Management, License management, User management Multicast management, Meeting management, Client Installer management. Whenever a new customer purchases license SuperAdministrator will create one User with GroupAdministrator role for the group. GroupAdministrator may create the other users with User role. GroupAdministrator also may create meeting by selecting the host, participant and Multicast IP (for multicast and unicast-multicast meeting types) for the group. GroupAdministrator may edit and delete users and meetings of his/her group. Role user belongs to a particular group. User may create meeting by selecting the host, participant and Multicast IP (for both unicast / multicast meeting types) from his/her group. User may also edit and delete meetings created by him/her. In this paper, role schema of SuperAdministrator (SA) is shown in **Figure 1**. Corresponding role schemas for GroupAdministrator (GA) and User are presented in [12].



Role Schema: SuperAdministrator (SA)

Description: This role is responsible for management (i.e. create, edit and delete) of group, license, user, multicast IP, meeting and client installer of the overall system.

Protocols and Activities: ManageGroup, ManageLicense, ManageMulticastIP, ManageClientInstaller, ManageUser, ManageMeeting

Permissions:

reads

license	*// to obtain information of license of different groups for creating user*
role	*// to obtain information of various roles to assign to user*
multicast IP	*// to obtain information of available multicast IP to assign to multicast meetings*
user	*// to obtain information of users to make host or participants of meetings*

generates / changes

group	*// creates, deletes and modifies groups*
license	*// creates, deletes and modifies license for groups*
multicast IP	*// creates, deletes and modifies to assign to multicast meetings*
client installer	*// creates, deletes client installers to be downloaded by users*
users	*// creates, deletes and modifies of different groups*
meeting	*// creates, deletes and modifies meetings to be joined by users*

Responsibilities

Liveness:

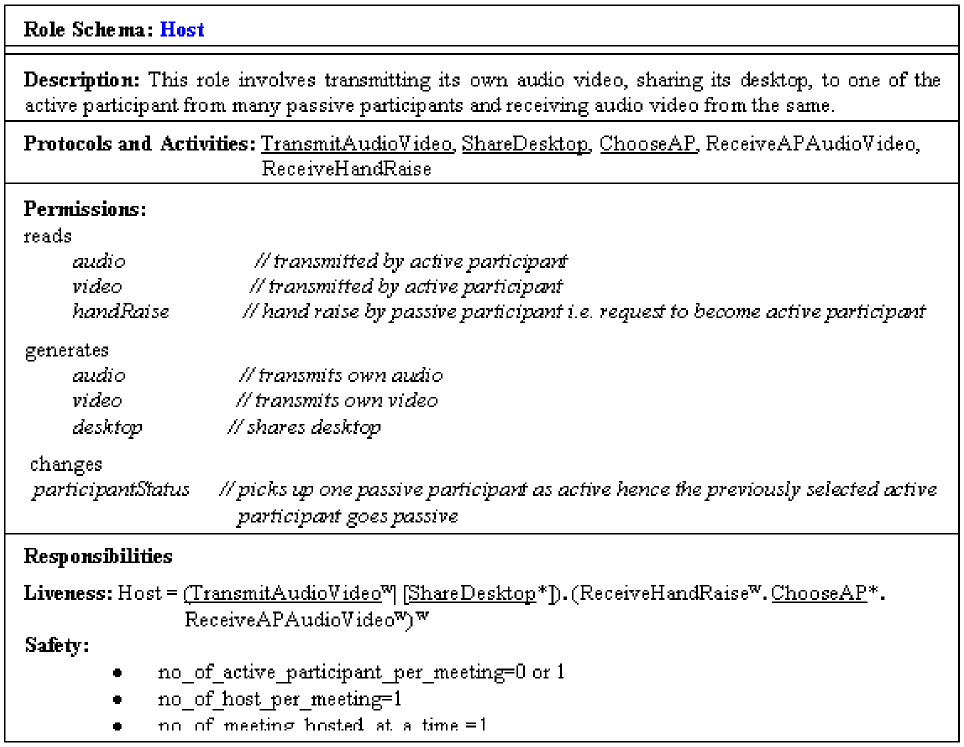
SuperAdministrator= $(\underline{\text{ManageClientInstaller}})^{w}$ | $((((\underline{\text{ManageGroup}}^{w}.\underline{\text{ManageLicense}}^{w}).\text{ManageUser}^{w})$ | $\underline{\text{ManageMulticastIP}}^{w}).\text{ManageMeeting}^{w})^{w}$

Safety:

- no_of_user_per_group <= no_of_license_for_the_group
- created_or_edited_user_role ≠ SuperAdministrator
- no_of_role _per_user = 1
- multicast_IP_for_multicast_meeting = unique
- version_no_for_client_installers = unique

Figure 1: Role schema of SuperAdministrator (SA)

Roles of *Meeting Executor* consist of Host, Active Participant, Passive Participant and Transponder. Host is selected at the time of meeting creation, though it can be changed by editing the meeting at any point of time. When a meeting starts there is Host and other passive participants present in the meeting. All passive participants receive host's audio video. In addition host can also share desktop. Passive participants may raise hand to become active. One of the passive participants is selected as active by the host. ActiveParticipant does interactive communication with the host. If any participant previously selected as active, it will be turned passive again automatically, if host selects another participant as Activate participant. One of the passive participants is selected as active by the host. Passive participant receives audio video from host as well as active participant, receives desktop sharing from host. It cannot transmit audio video. But it can raises hand to become active. Transmitted audio video, desktop sharing and hand raise may reach directly to the recipient or via n number of hops in-between depending upon the distance between the sender and receiver. There will be only one Transponder per network. Transponder is essentially a client agent playing role of Host, AP or PP in the same meeting. Transponder may work independently or collaborates with other Transponders. It receives audio video from Host and ActiveParticipant, as well as desk top sharing (DS) and hand raise of PassiveParticipant. It also capable of retransmit all these at the same time. In this paper, the role schema for Host is shown in **Figure 2**. Role schemas for ActiveParticipant (AP), Passive participant (PP) and Transponder (TP) are depicted in [12].


Role Schema: Host

Description: This role involves transmitting its own audio video, sharing its desktop, to one of the active participant from many passive participants and receiving audio video from the same.

Protocols and Activities: TransmitAudioVideo, ShareDesktop, ChooseAP, ReceiveAPAudioVideo, ReceiveHandRaise

Permissions:
reads
audio *// transmitted by active participant*
video *// transmitted by active participant*
handRaise *// hand raise by passive participant i.e. request to become active participant*

generates
audio *// transmits own audio*
video *// transmits own video*
desktop *// shares desktop*

changes
participantStatus *// picks up one passive participant as active hence the previously selected active participant goes passive*

Responsibilities

Liveness: Host = (TransmitAudioVideo^{w}| [ShareDesktop*]).(ReceiveHandRaise^{w}.ChooseAP*. ReceiveAPAudioVideo^{w})^{w}

Safety:
- no_of_active_participant_per_meeting=0 or 1
- no_of_host_per_meeting=1
- no_of_meeting_hosted_at_a_time =1

Figure 2: Role schema of Host

Interaction model follows the role model. The functionalities, activities and responsibilities of the roles are described in role model. Interaction model depicts the interaction between the roles on the basis of protocol. It also describes the characteristics and dynamics of each protocol. So it completely defines the protocol through which the roles will interact in the organization. Interaction model of *VCS* is shown in **Figure 3**.

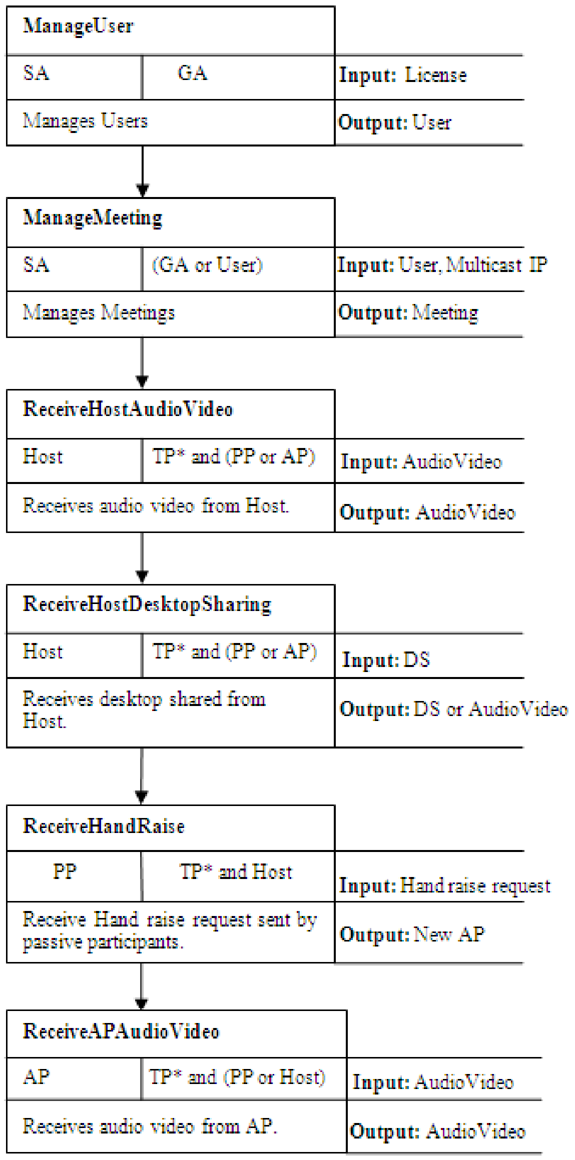


Figure 3: Interaction model of protocol Manage User and Manage Meeting

Detailed Design Phase

According to Gaia, the detailed design commences just after development of the role model and interaction model. In the detailed design phase, actually one-to-one correspondence between roles and agent classes is made to create agent model. It is also possible to combine closely roles into one agent. This is advantageous because of less number of classes and instances eventually reduce conceptual complexity.

However, this has to be done without:

- ❑ affecting the organizational efficiency,
- ❑ violating organizational rules and
- ❑ Creating problems (that is, without exceeding the amount of information it is able to store and process in a given amount of time).

Detailed design phase actually develops agent model and service model. In case of agent model, agent is a software program or class which would be implemented during implementation stage. In *Meeting Provider* sub organization, each role is mapped to its corresponding agent, while, group of roles are mapped to a single agent in *Meeting Executor* sub organization. Agent model of *Meeting Provider* and *Meeting Executor* are depicted in Figure 4 and Figure 5 respectively. In case of service model, the specific services of agents are identified. There are four properties related to each service, namely: input, output, pre-condition and post-condition. Input-output are actually derived from the interaction model or protocols, while pre and post conditions are restrictions on the execution and completion of the services respectively.

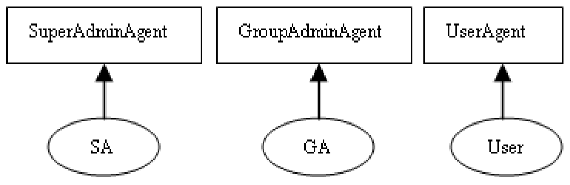


Figure 4: Agent Model of Meeting Provider sub organization

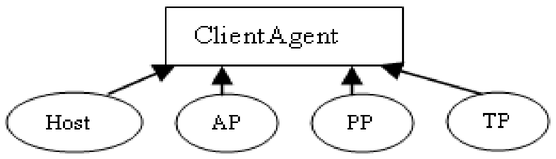


Figure 5: Agent Model of Meeting Executor sub organization

These services are derived from the protocols, activities and liveness properties of the roles that the agent implements. As a rule of tomb, there will be one service for each parallel activity of execution that the agent has to execute. The services model of VCS is represented in Table 2.

Table 2: Services Model of Video Conferencing Solution

Service	Input	Output	Pre-condition	Post-condition
ManageUser	License	Users	Number of user less than number of license for that group	Role of user is User or GA
ManageMeeting	User, Multicast IP	Meeting	Unique Multicast IP required	Meeting with one host and multiple participant
ReceiveHostAudioVideo	Audio Video	Audio Video	Host has to join meeting	Number of Host is exactly 1 for a meeting
ReceiveAPAudioVideo	Audio Video	Audio Video	Host has select AP	Number of AP is exactly 1 for a meeting
ReceiveHostDesktopShare	Desktop sharing (DS)	Desktop sharing (DS) or Audio Video	Host has to join meeting	Number of Host is exactly 1 for a meeting
ReceiveHandRaise	Hand raise request	New AP	Host and PP has to join meeting	

Next section presents agent interaction with the help of use case diagram.

4. AGENT INTERACTIONS IN VCS

Although according to Gaia methodology, agent interactions in *VCS* is shown through interaction model, but for ease of user friendliness UML technology need to incorporated in the design phase. Moreover, use case diagrams are playing major role in industry. So according to UML technology Use case diagrams for two sub-organizations are presented in this section. In order to so, it is mandatory to identify the actors and use cases or methods through which actors or external entity will interact with the *VCS* system.

In case of both sub organization, each role is mapped into corresponding actors and activity or protocols are identified as use cases. Hence, SuperAdministrator, GroupAdministrator and User are the actors in *Meeting Provider* sub organization, while Host, ActiveParticipant and PassiveParticipant are the actors in *Meeting Executor* sub organization. However, Transponder is essentially a client agent playing role of Host, ActiveParticipant or PassiveParticipant in the same meeting. So being a role in *Meeting Executor* sub organization, Transponder is not mapped as actors. **Figure 6** and **Figure 7** present use case diagrams for both sub organization respectively. Only major use cases are depicted in these diagrams.

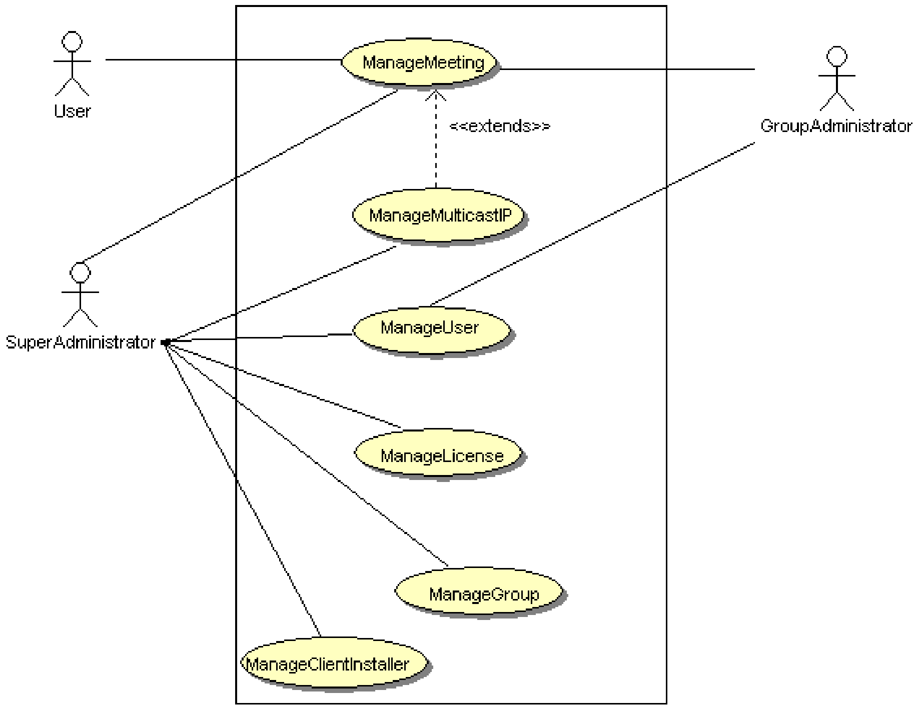


Figure 6: Use case diagram of Meeting Provider

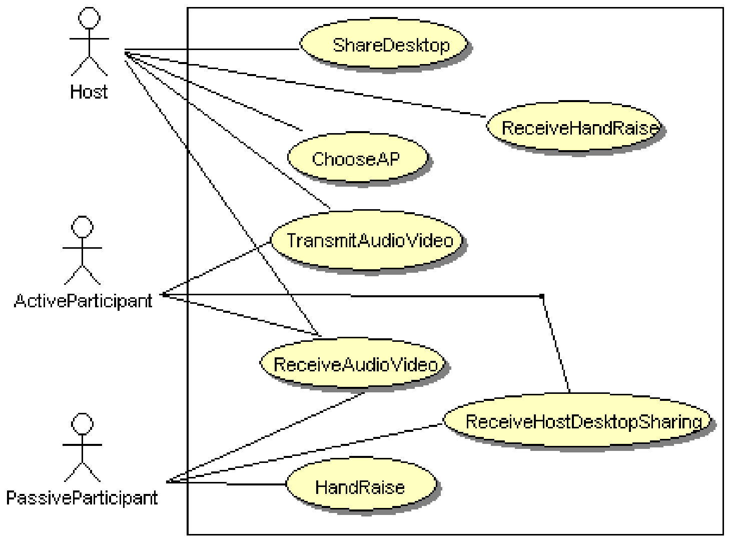


Figure 7: Use case diagram of Meeting Executor

5. Implementation Details

The video conferencing solution has been implemented using Java RMI technology. Java uses method invocation as its major means of communication. Java Agent Development Framework (JADE) is used as the platform for this initial implementation of *VCS*. Thus, Jade is an agent middleware, which implements an agent platform. JADE includes Java classes to support the development of applications agents, and the "run-time" environment that provides the basic services for agents to execute. It is a pure Java, middleware platform intended for the development of distributed multi-agent applications based on point-to-point communication. Moreover, it provides standard agent technologies keeping run-time overheads low.

For experimental purpose the environment is set up as following:

1.	Operating System	: Windows XP, Windows Vista, Windows7, & Ubuntu Linux 10.04
2.	Processor	: 2.8 GHz Intel Dual Core Processor (or equivalent) or higher
3.	Memory	: 1 GB RAM (Min)
4.	Disk Space	: 1 GB of free hard disk space
5.	Network Card	: 10/100 base T Network Interface Card
6.	Sound Card	: Built-in motherboard
7.	Camera	: PTZ camera or Webcam
8.	Drivers	: Latest Audio and Video drivers in windows

9. Audio Headset and Microphone or Headphone with Audio/Speaker output or USB speaker phone

Execution of *VCS* is done with the above-mentioned environment setup. **Figure 8** and **Figure 9** presents sequence diagram for both the sub organization, *Meeting Provider* and *Meeting Executor* respectively. These sequence diagrams actually show real life interaction between the different agents during conferencing.

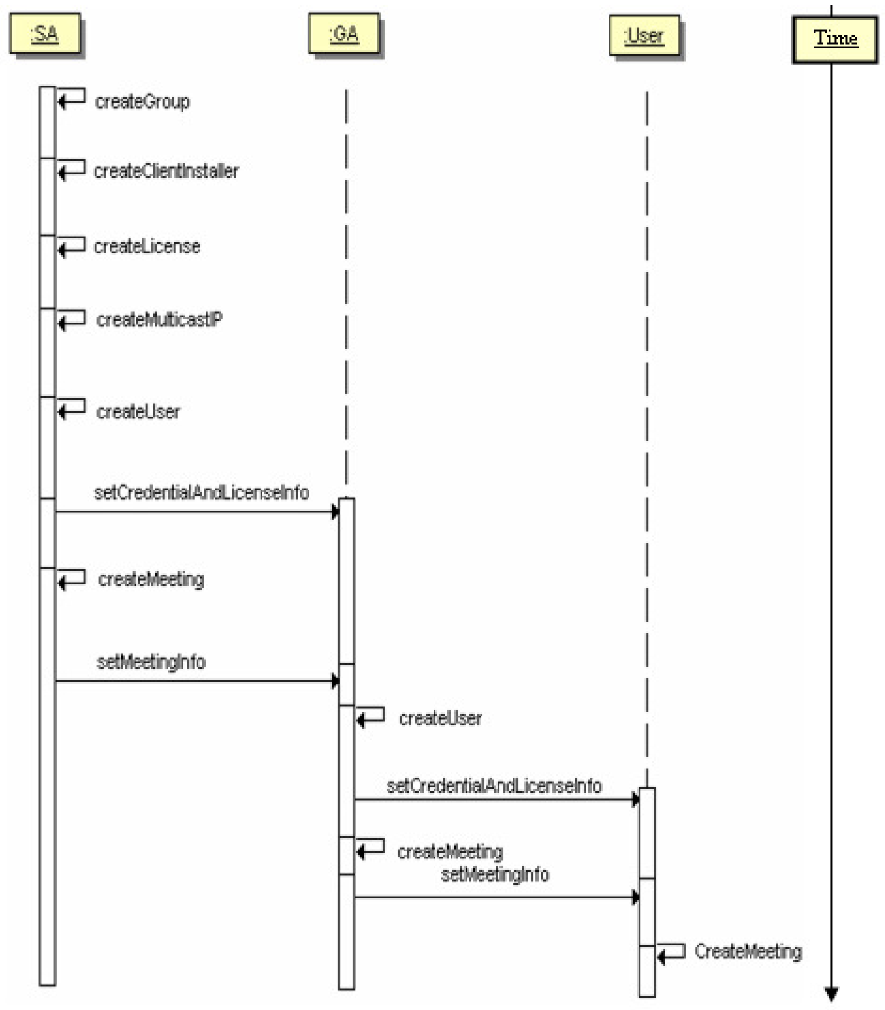


Figure 8: Sequence diagram of Meeting Provider

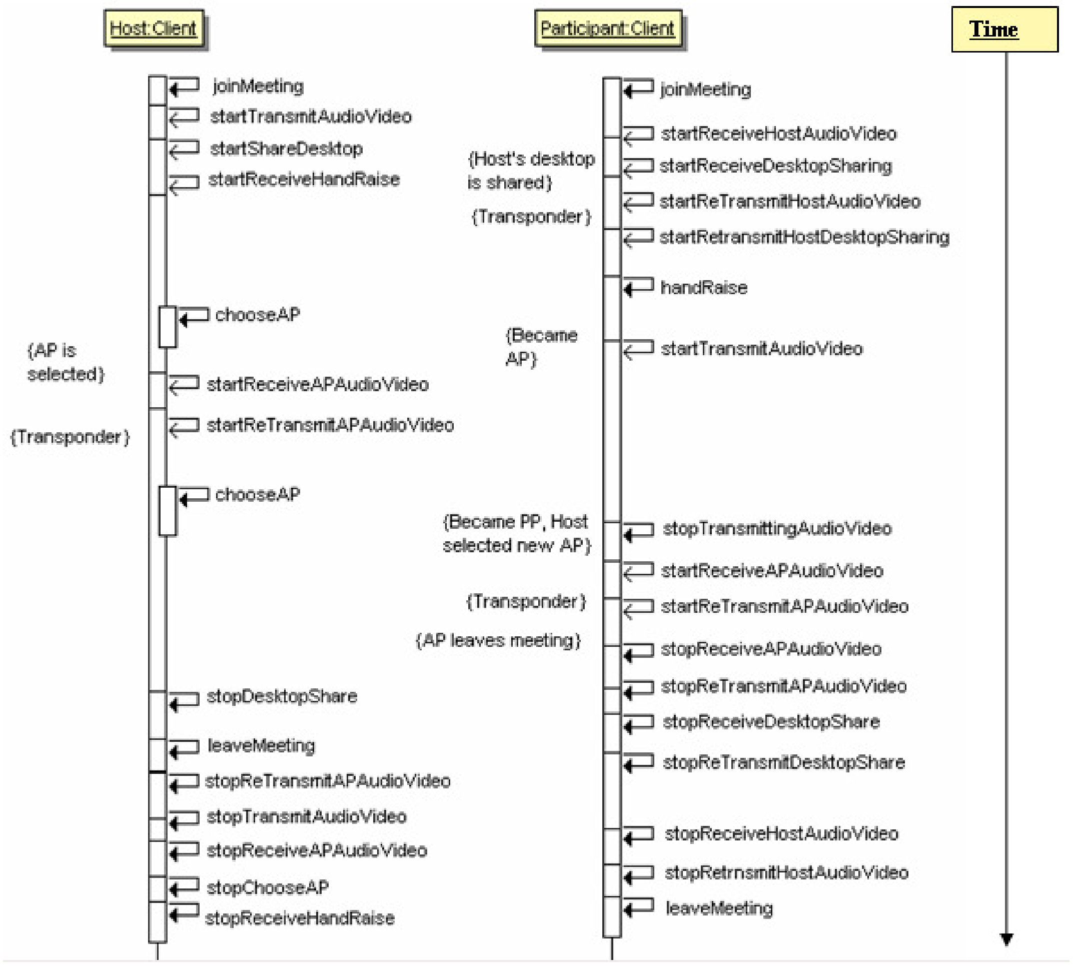


Figure 9: Sequence diagram of Meeting Executor

6. CONCLUSION

This paper clearly portrays the design of a video conferencing solution based on multi-agent system. This solution is capable of setting up live meetings between a host and a number of participants. *VCS* is a collection of autonomous interacting software agents can offer services in grid environment through deployment of these interconnected agents onto the resources. In order to do design and analysis of *VCS,* a software engineering approach using Gaia Methodology is used. The *role model*, *interaction model*, *agent model* and *services model* are discussed and developed in this research work. The current work demonstrates individual and autonomous agent structure as well as illustrates the interaction of the agents in proposed *VCS*. This paper also presents implementation details of the *VCS* with the FIPA-compliant agent platform, Jade framework. The main intention of future work is to achieve *VCS* in grid environment through collaboration of a few interacting agents.

REFERENCES

[1] Dimdim Video Conferencing Solutions- available from: www.dimdim.com

[2] GoToMeeting Video Conferencing Solutions- available from: www.gotomeeting.in

[3] PlaceCam Video Conferencing Solutions- available from: www.placecam.com

[4] Skype Video Call Solutions- available from: www.skype.com

[5] Vennfer Video Conferencing Solutions- available from: www.intellisysin.com

[6] Woovoo Video Conferencing Solutions- available from: www.oovoo.com

[7] Cernuzzi, L., Cossentino, M., Zambonelli, F., "Process Models for Agent- based Development", Journal of Engineering Applications of Artificial Intelligence, Vol. 18, No.2, (2005) 205-222.

[8] Cooper C. "Multi-Site Videoconferencing for the UK e-Science Programme, A Roadmap for the Future of Videoconferencing within e-Science, ",(UKERNA / Oxford Brookes University) and Michael Daw (University of Manchester), 2002

[9] Koutsabasis P., Darzentas J. S., Spyrou T., Darzentas J., "Facilitating User–System Interaction: the GAIA Interaction Agent Proceedings", 32nd Hawaii International Conference on System Sciences – 1999

[10] L. Rodriguez, A. Hume, L. Cernuzzi, E. Insfrán, "Improving the Quality of Agent-Based Systems: Integration of Requirements Modeling into Gaia", 2009 Ninth I International Conference on Quality Software 1550-6002/09 $26.00 © 2009 IEEE DOI 10.1109/QSIC.2009.43 278 2009 Ninth International Conference

[11] Moraitis, P and Spanoudakis, N, "The Gaia2Jade Process for Multi-Agent Systems Development", Applied Artificial Intelligence, 20: 2, pp: 251 — 273, 2006.

[12] Sen S. and A. De Sarkar, "Design of Video Conferencing Solution in Grid" in the Proceedings of the Second International Conference on Advances in Computing and Information Technology (ACITY 2012), July 13-15, 2012, Chennai, India, Volume -2, pp:445-457.

[13] Uyar A., W. Wu, H. Bulut and G. Fox, "An Integrated Videoconferencing System for Heterogeneous Multimedia Collaboration", Department of Electrical Engineering and Computer Science, Syracuse University Community Grid Lab, Indiana University, 2003.

[14] W. Huang, E. El-Darzi and Li Jin, "Extending the Gaia Methodology for the Design and Development of Agent-based Software Systems", 31st Annual International Computer Software and Applications Conference(COMPSAC 2007) 0-7695-2870-8/07 $25.00 © 2007 IEEE

[15] Wooldridge M, N. R. Jennings, and D. Kinny, "The Gaia Methodology for Agent-Oriented Analysis and Design", In Journal of Autonomous Agents and Multi-Agent Systems, vol. 3, no. 3, pp. 285-312, 2000.

[16] Zambonelli F, N. R. Jennings and M. Wooldridge, "Developing Multiagent Systems: The Gaia Methodology", ACM Transactions on Software Engineering Methodology, vol. 12, no. 3, pp. 317-370, July 2003.

AUTHORS

Ajanta De Sarkar is working as Associate Professor in the department of CSE in Birla Institute of Technology, Mesra. She is having altogether 16 years of experience including 6 years of Industry experience. Having graduated from Bethune College, University of Calcutta in B.Sc. (Mathematics) in 1993, and obtained MCA degree in 1996 from Jadavpur University. She has been awarded PhD in Engg. from Jadavpur University in 2009. Her field of Specialization is Distributed Computing, specifically Grid Computing. Her focused research area includes Grid Computing, Cloud Computing and Wireless Sensor Network.

Soumilla Sen is a Software Analyst and having 8 years of experience in Java JEE relevant Software Development. He obtained DOEACC B level degree in 2003 and MBA (IT) from Sikkim Manipal Unversity in 2007. Currently he is pursuing M. Tech in Birla Institute of Technology, Mesra since 2010.