Barreto, C. G., Filippo, D., Fuks, H. & Lucena, C.J.P. (2005) "*Integrating MAS in a Component-Based Groupware Environment*", AOSE-2006@AAMAS Agent-Oriented Software Engineering Workshop in AAMAS-International Joint Conference on Autonomous Agents & Multi-Agent Systems, Hakodate, Japan, May, 8-12, pp. 145-156.

Integrating MAS in a Component-Based Groupware Environment

Celso Gomes Barreto, Denise Filippo, Hugo Fuks and Carlos José Pereira de Lucena

Software Engineering Laboratory (LES) - Computer Science Department
Catholic University of Rio de Janeiro (PUC-Rio)
{celsogbj, denise, hugo, lucena}@inf.puc-rio.br

Abstract. When developing groupware, the designer must be mindful of the need for the solution to be sufficiently adaptable to meet the needs of different kinds of work groups as well as the evolution of work processes. This environment of shifting demands favors a component-based development approach. However, components are incapable of providing the most suitable level of abstraction in cases requiring software artifacts with properties such as autonomy, pro-activity, reactivity and social ability. In these cases, software agent technology and multi-agent systems are more suitable. This article demonstrates how groupware developed with a component-based architecture can benefit from the introduction of an agent container. The case study is based on the AulaNet Learning Management System and, as proof of concept, a multi-agent system is implemented in which agents in mobile devices interact with agents on an AulaNet server.

1 Introduction

Groupware is software that supports the interaction between individuals who work to achieve a common objective. Learningware, meanwhile, refers to groupware that supports collaborative teaching/learning. A groupware solution involves multidisciplinary aspects in its construction and is difficult to apply and test, being especially vulnerable to failures [1]. Each group has its own needs and these alter over time, making it essential for groupware to be developed in an iterative manner and that the experience of using it informs towards its evolution, creating new services or improving the functionalities of existing services. By employing a component-based architecture, groupware can be adapted by plugging and unplugging components, producing an application that better matches the needs of a work group.

Although components are an elegant solution to the adaptability of groupware solutions, they fail to provide the most suitable level of abstraction for the creation of software artifacts with characteristics such as autonomy, pro-activity, reactivity and social abilities. These characteristics enable the creation of services that facilitate collaborative processes. Examples of services with these properties are: the capacity of autonomy enables the creation of "Virtual Tutors," responsible for suggesting reading material to a student based on his study record [2]; pro-activity and reactivity allow the groupware to inform the course mediators of any undesirable conditions occurring in a learningware solution, such as a lengthy period of student inactivity, for example [3]; social abilities are used by personal assistants, which can schedule meetings between group members [4]. Such characteristics are intrinsic to the multi-agent system (MAS) paradigm.

This article demonstrates how a groupware solution using a component-based architecture can benefit from integration with an agent framework, JADE [5], using a real world example, AulaNet. Section 2 describes AulaNet and the 3C Model on which it is based. Section 3 presents the AulaNet architecture. Section 4 shows how JADE is integrated with the AulaNet architecture, enabling the use of agents. Section 5 presents a proof of concept based on an MAS developed to validate the integration and, finally, Section 6 presents our conclusions and outlines future work.

2 The AulaNet Learning Management System

The AulaNet is an example of web-based learningware, a type of groupware designed for collaborative learning that has been developed in the Software Engineering Laboratory at Catholic University of Rio de Janeiro since 1997. Available in Portuguese, English and Spanish, the AulaNet can be downloaded for free from EduWeb (www.eduweb.com.br), a company that customizes this groupware for its different clients. Currently, the AulaNet is used in dozens of companies and universities around the world. Recently, the development of an extension of the AulaNet for mobile devices, called AulaNetM, was also started [3].

The approaches used in the development and classification of the AulaNet's services are based on the refinement of the 3C model [1] proposed by Ellis, Gibbs & Rein [6]. According to this model, depicted in Figure 1, for a group's objectives to be attained, it needs to communicate, which generates commitments. In order to guarantee that these are fulfilled and to deal with any conflicts that may arise, it is essential that the commitments generated by the communication are coordinated. Once coordinated, group members can cooperate, operating in conjunction on a shared workspace and undertaking the collaborative work. Cooperation demands communication and the cycle starts again. The participants receive feedback from their actions and feedthrough of the actions of their peers through perceptual information related to interaction with other participants.

The 3C Model is frequently used to classify collaborative systems [7]. The AulaNet's services can be classified according to this model [1]. These services enable teachers to use the AulaNet both as a complement to classroom-based courses and as entirely long-distance courses.

The AulaNet development team also runs the Information Technology Applied to Education (ITAE) course [8], offered every semester to undergraduate and graduate students in the Computer Science Department of this university since 1998. This course is run entirely at a distance using the AulaNet environment and the experience obtained with it allows continual improvement of the AulaNet, and vice-versa.

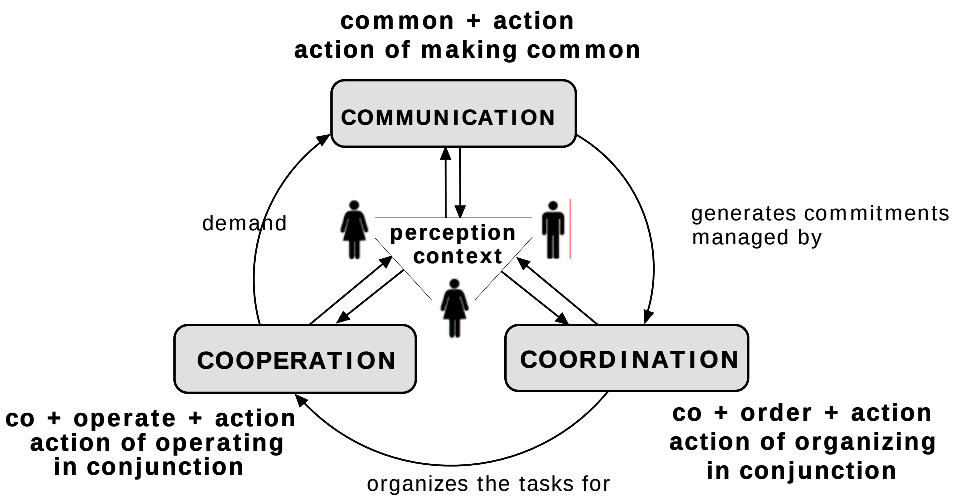


Fig. 1. 3C Collaboration Model

During the design and development of groupware such as the AulaNet, the designer of the groupware solution has to be mindful that the application may eventually be used by a variety of groups, each with its own work methodology, and that these characteristics may evolve over time. The groupware must be adaptable enough to allow the composition of groupware solutions that meet the needs of a specific group. This scenario favors a component-based development approach [1]. Below we describe the architecture of version 3.0 of AulaNet.

3 The Architecture of AulaNet 3.0

The AulaNet must be configurable by its administrators. Each course has its own pedagogical approach, depth of content, and so on. It is impossible to foresee all the needs of users and, even if it were possible, the environment would become difficult to use and configure. As we saw earlier, this scenario of shifting requirements favors a component-based development approach, providing greater flexibility in meeting these needs [9].

The AulaNet 3.0 architecture uses component frameworks [10] that offer the execution environment and the contracts that need to be satisfied by the implementation of the components. The figure below depicts the AulaNet 3.0 architecture. There are 2 levels of componentization: the groupware service level and the collaboration component level.

Services are plugged and unplugged into and from the Groupware Component Framework in order to assemble a customized environment for each group [9]. This component framework provides the contracts that the groupware services – for example, Debate and Conference – must implement in order to be plugged into the AulaNet architecture. Meanwhile, services are built using collaboration components, which must implement the contracts defined in the Collaboration Component Framework. For instance, the main components used in the Conference service are the Message Manager and the Message Locator.

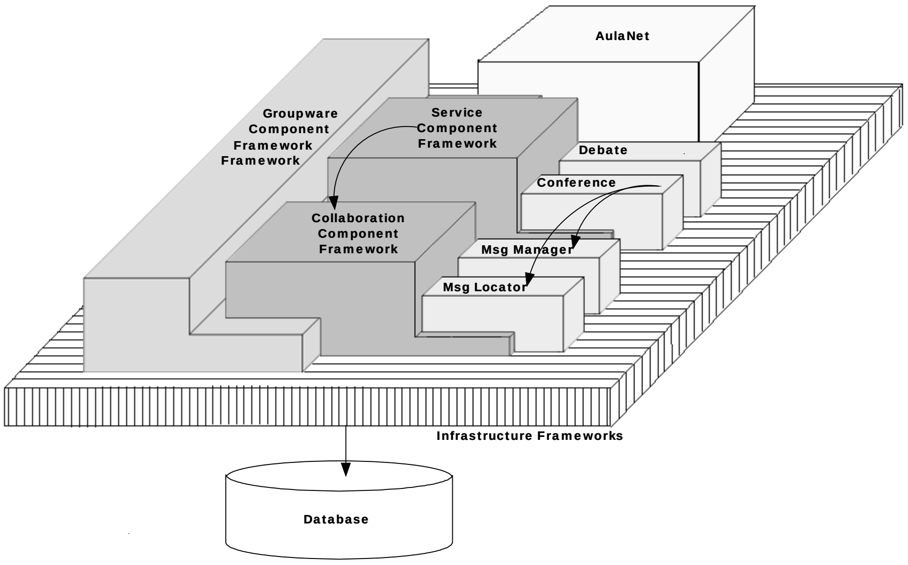

AulaNet
Groupware Component Framework Framework
Service Component Framework
Debate
Conference
Collaboration Component Framework
Msg Manager
Msg Locator
Infrastructure Frameworks
Database

Fig. 2. AulaNet 3.0 Architecture

Groupware component designers and developers face innumerable problems. As well as understanding about collaboration, they have to deal with infrastructural issues such as data persistence, transaction control, security and so forth. Groupware development is already a difficult task in itself due to its multidisciplinary nature and the heterogeneity of the different target work groups: it should not, therefore, be complicated by low-level issues. To deal with the latter issues, the architecture depends on a base of infrastructural services comprised of application frameworks [11].

The Hibernate framework [12] is used to ensure the persistence of data within the system's database. This framework deals with complex questions arising from the differences between the Object-Oriented and Relational paradigms. The application’s user interface is built using the JavaServer Faces framework [13]. This framework provides an implementation of the MVC architectural standard [14] and offers means for creating and reusing interface components. The Spring framework [15] is used to configure the dependence between application components. This framework promotes loose coupling between components, configuring them through Dependency Injection [16], where the dependences between frameworks are declared in a descriptor file and automatically configured by the framework. In addition, Spring provides services such as transaction management, security and remote service display.

Software components enable the building of groupware solutions that satisfy the specific needs of a work group. However, they fail to supply the most suitable degree of abstraction for dealing with some aspects of collaborative learning. The mediators could be notified that a conference is not proceeding as planned and therefore take some action to animate it. The system could offer courses and content based on the student's study history. A debate session could be animated by an intelligent software solution. These and other features can be implemented through the use of characteristics like autonomy, pro-activity, reactivity and social ability, typically found in multi-agent systems (MAS) [17].

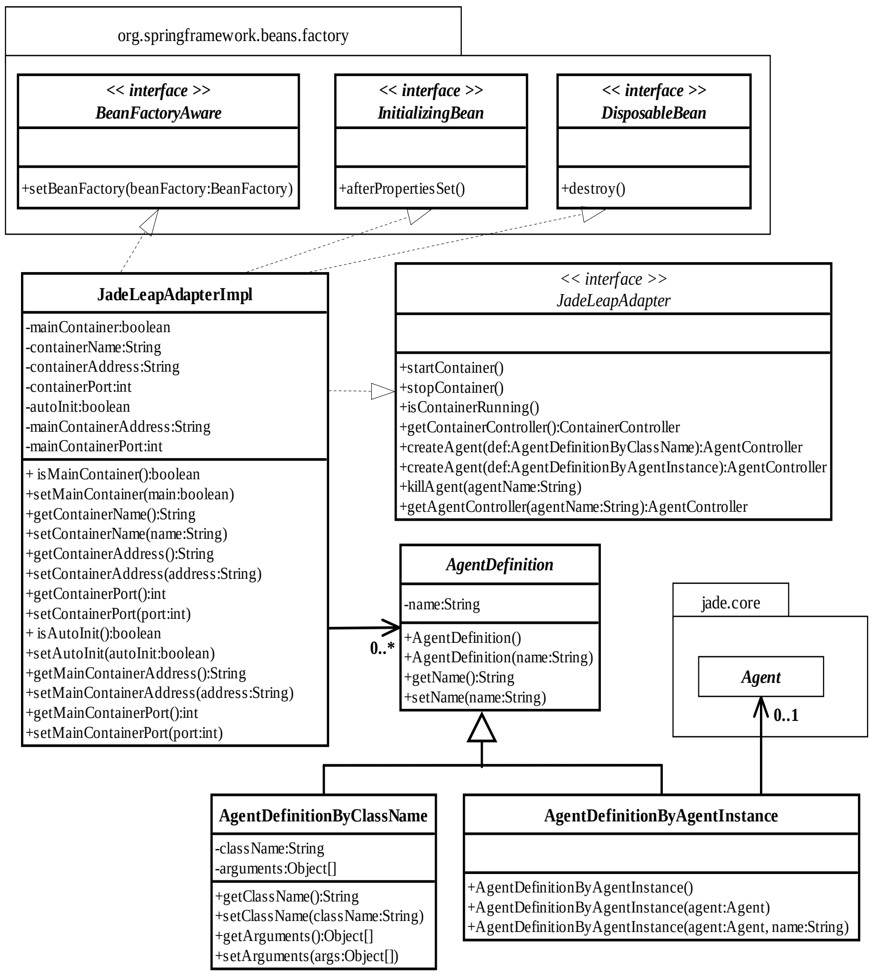
4 Integrating JADE with AulaNet

JADE is an object-oriented application framework [11][18] geared towards the development of multi-agent systems in compliance with specifications established by FIPA (Foundation for Intelligent Physical Agents) [19]. FIPA is an organization recognized by the IEEE which promotes agent-based technologies and the interoperability of their specifications with other technologies. JADE was chosen since it is a mature agent framework compatible with FIPA specifications, used commercially by some companies such as Telecom Italia LAB, Whitestein Technologies AG and Acklin B.V. [26], written in Java and enables the use of agents in either mobile devices or desktop servers through the LEAP extension package [27].

In addition to the agent development framework, JADE includes an execution environment for agents, called a container, and a graphic tool where the state of agents is administered and monitored. A set of JADE containers, executed on the same machine or on remote machines, makes up part of a platform. Each platform must have a main container where the peripheral containers are registered. When executed in mobile devices, containers can also make use of either the split option, where the processing of the container is divided with a server machine, or stand-alone, where a complete container is executed on the device.

In order to incorporate JADE or any other framework into AulaNet's business layer, it needs to be compatible with Spring. The latter is assigned the task of configuring the dependencies between components and, in a way, is a core element in the AulaNet 3.0 architecture. As Spring does not offer native support for integration with JADE, it was necessary to develop an Adapter [20]. It enables the JADE agent container to be started within an application that uses Spring and also allows JADE-created agents to call component methods defined in Spring. The dependency of agents and components is configured through Dependency Injection. Adapter classes are shown in the diagram depicted in Figure 3.

The Adapter also allows AulaNet components to create or destroy agents or access existing agents, adding new behaviors to them. The AgentDefinition class and its subclasses are used to define an agent. These definitions are used by the Adapter to create agents and add them to the JADE container. The abstract class AgentDefinition uses only the name of the agent to define it, and is not sufficient to add an agent to the JADE container, hence one of its subclasses needs to be used. The AgentDefinition-ByClassName subclass is used to define an agent based on the name of the class that implements the agent and the arguments that must be sent to it. The AgentDefinition-ByAgentInstance subclass defines an agent through an instance of the jade.core.Agent class.



org.springframework.beans.factory
<< interface >> BeanFactoryAware
+setBeanFactory(beanFactory:BeanFactory)
<< interface >> InitializingBean
+afterPropertiesSet()
<< interface >> DisposableBean
+destroy()
JadeLeapAdapterImpl
-mainContainer:boolean
-containerName:String
-containerAddress:String
-containerPort:int
-autoInit:boolean
-mainContainerAddress:String
-mainContainerPort:int
+ isMainContainer():boolean
+setMainContainer(main:boolean)
+getContainerName():String
+setContainerName(name:String)
+getContainerAddress():String
+setContainerAddress(address:String)
+getContainerPort():int
+setContainerPort(port:int)
+ isAutoInit():boolean
+setAutoInit(autoInit:boolean)
+getMainContainerAddress():String
+setMainContainerAddress(address:String)
+getMainContainerPort():int
+setMainContainerPort(port:int)
<< interface >> JadeLeapAdapter
+startContainer()
+stopContainer()
+isContainerRunning()
+getContainerController():ContainerController
+createAgent(def:AgentDefinitionByClassName):AgentController
+createAgent(def:AgentDefinitionByAgentInstance):AgentController
+killAgent(agentName:String)
+getAgentController(agentName:String):AgentController
AgentDefinition
-name:String
+AgentDefinition()
+AgentDefinition(name:String)
+getName():String
+setName(name:String)
0..*
jade.core
Agent
0..1
AgentDefinitionByClassName
-className:String
-arguments:Object[]
+getClassName():String
+setClassName(className:String)
+getArguments():Object[]
+setArguments(args:Object[])
AgentDefinitionByAgentInstance
+AgentDefinitionByAgentInstance()
+AgentDefinitionByAgentInstance(agent:Agent)
+AgentDefinitionByAgentInstance(agent:Agent, name:String)

Fig. 3. Diagram of Adapter Classes

The JadeLeapAdapter interface defines the methods of the Adapter responsible for integrating JADE-LEAP and Spring. Methods are defined for starting the JADE container (startContainer()), stopping execution of the container (stopContainer()), checking whether the container is running (isContainerRunning()) and recovering control of the container (getContainerController()): these allow more complex operations to be performed in the JADE container. Other methods exist for creating an agent, according to description by class name or by instance (createAgent()), destroying an agent (killAgent()) or recovering the controller of the agent used to perform operations such as adding behaviors to agents.

Implementation of the JadeLeapAdapter interface is performed by the JadeLeapAdapterImpl concrete class. The latter also implements Spring's BeanFactoryAware, InitializingBean and DisposableBean interfaces. The first defines the setBeanFactory() method, which is called by Spring to send to the BeanFactory. Through this factory, other beans defined in Spring's configuration file can be recovered by the agents created through definition by class names. Agents created through definition by agent instances do not require the factory since they can access other beans defined in Spring through the Dependency Injection mechanism. The InitializingBean interface defines the afterPropertiesSet() method called when the Adapter is started and is used to start the JADE container automatically. Finally, the DisposableBean interface defines the destroy() method, which is called shortly before the Adapter is destroyed, used to stop the JADE container automatically.

Additionally, the JadeLeapAdapterImpl class defines properties that allow configuration of JADE's execution mode. The mainContainer property indicates whether the container to be executed is a main container. If this property is configured as false, the mainContainerAddress and mainContainerPort properties must be configured with the address and port of the main container. The containerName, containerAddress and containerPort properties are used to specify the name, address and port, respectively, of the container started by the Adapter. The agentDefinitions vector is configured with the agents created when the container is started and the autoInit property defines whether or not the container is started automatically when the Adapter is created.

However, the Adapter has some limitations. When migrating to another machine, the references an agent possesses for local services become inconsistent. The same happens with agents cloned on other machines. Furthermore, agents created through definition by class name are unable to use Spring's Dependency Injection, since they are instanced by JADE rather than Spring. To solve this problem, the Adapter adds a Spring BeanFactory in the last position of the argument vector sent to the agent. This allows the agent to recover instances of beans defined in Spring.

The Adapter must be configured in the Spring configuration file in order for integration to occur. As no API specific to JADE or JADE-LEAP is used in its construction, the latter can be used with both. The Adapter was not developed to support the split execution mode in mobile devices, since Spring was developed to be used only in J2SE and J2EE environments and therefore does not work in mobile devices. However, this does not prevent an agent running in a JADE container in a J2SE or J2EE environment with Spring from communicating with another agent in a MIDP [21] or a Personal Profile [22] J2ME [23] mobile device, as we shall see in the next section.

5 Proof of Concept: Mediator Assistant

A proof of concept was developed to test the integration between JADE and the AulaNet 3.0 architecture through Spring using Adapter. The developed application is used to test the applicability of agents in both the AulaNet and AulaNetM. The application's objective is to alert mediators whenever an undesirable situation is taking place within the Conference service. Four normally undesirable conditions are verified. The first relates to its activity level, which generates a notification whenever the application detects the absence of messages for a period of time greater than a predetermined interval. The second condition verifies whether a question is not receiving enough attention or whether it is polarizing the conference in detriment to other questions. In this case, mediators are notified whenever a message categorized as “Question” possesses a number of replies below or above a pre-set limit. The third information relates to the activity level of the students: a notification is generated when a student displays a very low level of involvement. The fourth and last information provides a measurement allowing mediators to stimulate the conference’s level of interactivity: for example, a notification is generated every time that the percentage of the tree structure leaves of the conference message is above 50% (when more than half of the messages have not yet received a reply) [24]. Due to the different characteristics of classes and different conference aims, the values of the parameters for the notifications to be issued should be configured by the mediators for each conference through an interface specifically designed for this purpose.

After receiving these notifications, mediators can take decisions that alter the evolution of the conference, overcoming the undesirable situations. AulaNetM provides means for verifying these situations through visual information as shown in Figures 4 and 5.

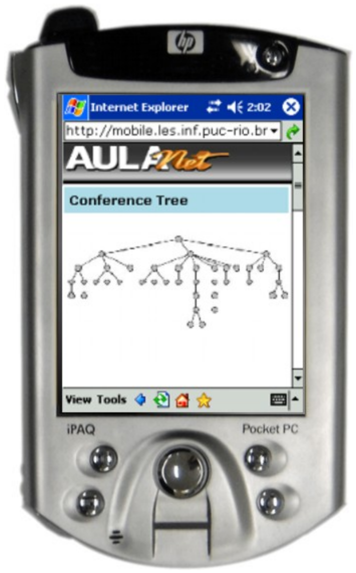
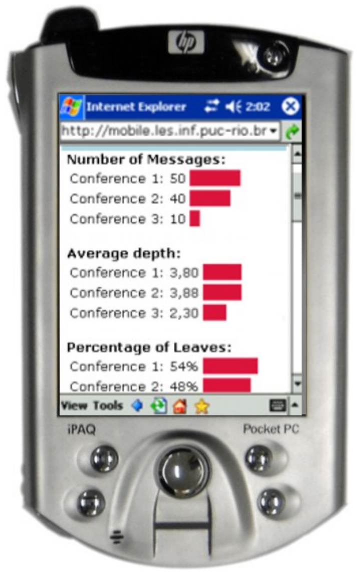


Fig. 4. Visual form of the tree structure on a conference



Number of Messages:
Conference 1: 50
Conference 2: 40
Conference 3: 10

Average depth:
Conference 1: 3,80
Conference 2: 3,88
Conference 3: 2,30

Percentage of Leaves:
Conference 1: 54%
Conference 2: 48%

Fig. 5. Partial statistics from three conferences an ITAE edition

The extraction of this information is not automated, i. e., mediators must directly access AulaNetM to obtain the information. Consequently, in most cases mediators become aware of the condition too late. In addition, interviewed mediators reported some cases of difficulty in establishing connection and maintaining it once established. In these cases, the mediators spent more time trying to connect or reconnect than actually analyzing the conference.

The developed MAS application solves these problems. Mediators no longer have to check their PDAs all the time, i.e., monitor the state of the conference, since the agents already take care of this and the fault tolerance mechanism present in JADE enables messages left undelivered due to connection failures to be postponed and delivered when the connection is re-established in a form that is invisible to the user. The application comprises a multi-agent system composed of two distinct types of agent: the Conference Agent and the Mediator Assistant.

The Conference Agent is executed within a JADE-LEAP container running on the AulaNet server. Through the Adapter, the Conference Agent is capable of accessing the AulaNet components through the dependencies configured by Spring. The Conference Agent is proactive and reactive and continuously monitors the AulaNet Conference in search of undesirable situations. Since each AulaNet course may contain several conferences, a Conference Agent is created upon the activation of each new conference. On deactivation, this agent is destroyed.

The Mediator Assistant is executed within a JADE-LEAP container ran within a PDA iPAQ 5555. The J2ME CDC CrEme virtual machine [25] is used to enable the execution of the application and the JADE container. The Mediator Assistant runs on the PDA the whole time, waiting for messages from the Conference Agent. On receiving a message, the Mediator Assistant issues an alert, notifying its user of the occurrence of the undesired situation. This agent is autonomous and can opt to not display an alert in determined cases, such as direct disabling by the user or excessive repetition of the alert. Figures 6 and 7 show the Mediator Assistant being executed.

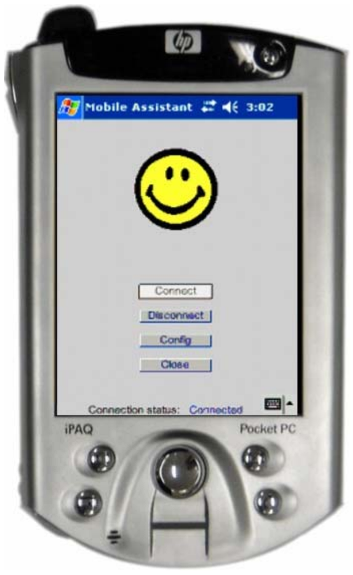
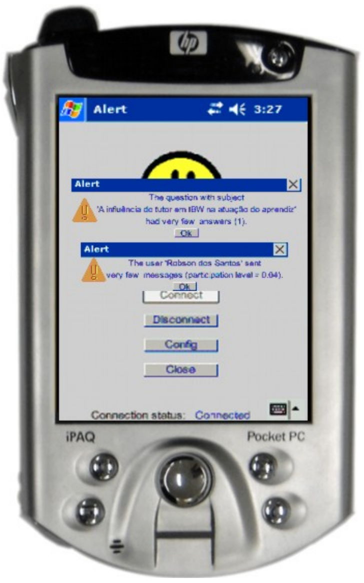


Fig. 6. Mediator Assistant Agent



The question with subject
'A influência do tutor em IBW na atuação do aprendiz'
had very few answers (1).

The user 'Robson dos Santos' sent
very few messages (participation level = 0.04).

Fig. 7. Mediator Assistant Alerts

Figure 6 shows the agent on standby, waiting for alerts. Figure 7 shows the agent after receiving many alerts. As there may be various Mediator Assistant agents interested in various conferences, a yellow pages service offered by JADE is used to keep records of agents interested in notifications. After being started, a Mediator Assistant agent publishes its interest in a particular conference on the yellow pages service. In turn, the Conference Agent, on perceiving an undesirable situation, consults the Yellow Pages service to obtain an updated list of the agents who should receive notification.

The application is executed in the background. When the agent receives an alert that should be displayed, it is brought into the foreground, a beep is played and the message is displayed. If the PDA is switched off, messages to the agent are postponed. The next time it is switched on, the JADE container re-establishes the connection with the main container and the agent receives all the previously undelivered messages.

Alerted by the Mediator Assistant agents, mediators become aware sooner of an undesirable situation, such as a lull in message traffic over a determined period of time. They can then take measures to try to improve or resolve these conditions.

6 Conclusions and Future Work

A groupware solution must be flexible enough for it to be adapted to the work processes of the variety of groups that use it and to accompany the evolution of the needs of these groups. This scenario of shifting requirements favors a component-based development approach. However, components lack the level of abstraction most suited to services that demand autonomy, pro-activity, reactivity and social abilities. These characteristics are commonly found in multi-agent systems (MAS). As a result, a project was started to integrate the AulaNet architecture with the JADE agent framework.

In order to be integrated with the AulaNet architecture, JADE needed to be integrated with the Spring framework. The latter’s tasks include configuring the dependence between components and also came to be used to configure the dependence between agents and components. As Spring does not provide native support for integration with JADE, an Adapter was developed. It enables the JADE container to be started from the AulaNet and allows JADE agents to call the AulaNet component methods. As a result, agents can be integrated without having to make substantial changes to the architecture and altering the application's other services.

The MAS was developed on the basis of a real problem in which course mediators need to be informed of the occurrence of undesirable conditions during conferences. By adding agents, the system becomes reactive, reacting to modifications in the environment (the conference), pro-active, since it takes the initiative and notifies the mediator as soon as the situation is detected, and autonomous, since it can choose which alerts to send to the user, protecting the mediator from being bombarded with identical messages.

Although this application signals an advance in tools providing support for conference moderating, it still possesses many limitations that will be dealt with in future work. First, the Conference Agent always sends messages to the Mediator Assistant agents registered in the yellow pages service. The latter, in turn, decides whether or not to notify the user. This frequently results in a waste of bandwidth and processing on the PDA, a device with limited resources. The most elegant solution would use an agent with mobility, which would migrate from the PDA to the AulaNet server and remain there until a critical situation is detected. A second limitation is that the agent only notifies mediators of an undesirable situation, when it could be more proactive and attempt to do something to remedy the situation. Other possibilities under study include sending alerts by alternate means such as e-mail and SMS.

In addition, future works will involve experimenting with other uses of agents in the AulaNet environment, including the Tutor Agent, Group Formation Agent, Moderator Agent and Notifier Agent. The Tutor Agent will be used to suggest reading content to students according to their course performances and their topics of interest. Meanwhile, the Group Formation Agent will be used to form work groups based on criteria such as the knowledge acquired by students, affinity, interests, etc. AulaNet's Debate service is used to conduct chat sessions. Usually, one student is elected session moderator and is responsible for conducting the debate session. The Moderator Agent can substitute the student in special conditions, when the latter is unable to perform the task of moderating. Finally, the Notifier Agent will be used to notify students and mediators whenever some event relating to the course occurs. This agent could be configured, for example, to notify the student whenever his or her message receives a reply during the conference.

The multi-agent system technology has the potential to solve some of the problems that arise during the development of groupware for which component technology fails to provide the best level of abstraction. This article demonstrated how the AulaNet, a groupware solution with a component-based architecture, benefits from integration with an agent container.

7 Acknowledgments

The AulaNet Project is funded by the Ministry of Science and Technology through the Multi-Agent Systems for Software Engineering project (ESSMA) award nº 552068/2002-0. It is also funded by individual awards from CNPq: Carlos Lucena nº 300031/92-0 and Hugo Fuks nº 301917/2005-1; CAPES: Celso Gomes Barreto.

References

1. Fuks, H., Raposo, A. B., Gerosa, M. A. and Lucena, C. J. P. Applying the 3C Model to Groupware Development, International Journal of Cooperative Information Systems (IJCIS), v.14, n.2-3, Jun-Sep 2005, World Scientific, ISSN 0218-8430, p.299-328
2. Marin, B. F., Hunger, A., Werner, S., Meila, S. and Schuetz, C. A Generic Framework for an Interface Tutor Agent within a Virtual Collaborative Learning Environment - Proceedings of the IEEE International Conference on Advanced Learning Technologies, ICALT 2004, 30 August - 1 September 2004, Joensuu, Finland. IEEE Computer Society 2004, ISBN 0-7695-2181-9
3. Filippo, D., Fuks, H. and Lucena, C. J. P. AulaNetM: Extension of the AulaNet Environment to PDAs. CONTEXT 2005, 5th International and Interdisciplinary Conference on Modeling and Using Context, Workshop 10: Context and Groupware, CEUR Workshop Proceedings, ISSN 1613-0073, vol. 133, Paris, France, 5-8 July, 2005.

4. Modi, P. J., Veloso, M., Smith, S. F. and Oh, J. CMRadar: A Personal Assistant Agent for Calendar Management. In 6th International Workshop on Agent-Oriented Information Systems (AOIS), pages 134-148, 2004.
5. Bellifemine, F., Caire, G., Poggi, A. and Rimassa, G. JADE – A White Paper. EXP in search of innovation Volume 3 - n. 3 - September 2003– Special issue on JADE. pp. 6-19.
6. Ellis, C.A., Gibbs, S.J. and Rein, G.L. Groupware - Some Issues and Experiences. Communications of the ACM, v. 34, n. 1 (1991) p. 38-58.
7. Borghoff, U.M. and Schlichter, J. H. Computer-Supported Cooperative Work: Introduction to Distributed Applications. Springer Press, USA, 2000
8. Fuks, H., Gerosa, M. A. and Lucena, C. J. P. The Development and Application of Distance Learning on the Internet. Open Learning - The Journal of Open and Distance Learning, Vol. 17, N. 1, ISSN 0268-0513, 2002 pp. 23-38.
9. Gerosa, M. A., Pimentel, M. G., Raposo, A. B., Fuks, H. and Lucena, C. J. P. Towards an Engineering Approach for Groupware Development: Learning from the AulaNet LMS Development. Proc. of the 9th International Conference on CSCW in Design (CSCWiD), vol. 1, p. 329-333. Coventry, U.K., May 2005. ISBN 1-84600-004-1.
10. Szyperski, C. Component software: beyond object-oriented programming. New York: ACM Press; Harlow: Addison - Wesley, c1998. 411 p. ISBN 0201178885 (enc.).
11. Fayad, M. E., Schimidt and D. C., Johnson, R. E. Building application frameworks: object-oriented foundations of framework design. New York: J. Wiley, c1999. 664 p. ISBN 0471248754, 1999.
12. King, G. and Bauer, C. Hibernate in Action. Manning Publishing Co., 2005.
13. Geary, D. and Horstmann, C. Core JavaServer Faces, Sun Microsystems, Inc, 2005.
14. Fowler, M. Patterns of Enterprise Application Architecture. Addison-Wesley, 2002.
15. Walls, C. and Breidenbach, R. Spring in Action. Manning Publishing Co., 2005.
16. Fowler, M. Inversion of Control Containers and the Dependency Injection pattern. Article available at http://martinfowler.com/articles/injection.html.
17. Wooldridge, M. and Jennings, N. R. Intelligent agents: theory and practice. Knowledge Engineering Review Volume 10 No 2, June 1995.
18. Bellifemine, F., Caire, G., Trucco, T and Rimassa, G. JADE Programmer's guide, 2005.
19. FIPA - http://www.fipa.org/.
20. Gamma, E., Helm, R., Johnson and R., Vlissides, J. Design patterns: elements of reusable object-oriented software. Massachusetts: Addison-Wesley, 1994. 395p. ISBN 0201633612 (Enc.).
21. MIDP - http://java.sun.com/products/midp
22. Personal Profile - http://java.sun.com/products/personalprofile
23. J2ME - http://java.sun.com/products/j2me
24. Gerosa, M. A., Pimentel, M., Fuks, H. and Lucena, C. J. P. Analyzing Discourse Structure to Coordinate Educational Forums. The 7th International Conference on Intelligent Tutoring Systems - ITS-2004, Maceió-AL, August 30th to September, 3rd 2004, Lecture Notes on Computer Science LNCS 3220, ISBN 3540-229485, ISSN 0302-9743, pp. 262-272.
25. CrEme - http://www.nsicom.com/Default.aspx?tabid=138.
26. JADE Home Page - http://jade.tilab.com/.
27. Caire, C. LEAP User Guide, 2005.