A Distributed Event Service for Adaptive Group Awareness

Dominique Decouchant[1], Ana María Martínez-Enríquez[2], Jesús Favela[3], Alberto L. Morán[1,4], Sonia Mendoza[1], and Samir Jafar[1]

[1] Laboratoire "Logiciels, Systèmes, Réseaux", Grenoble, France
{Dominique.Decouchant,Alberto.Moran,Sonia.Mendoza,Samir.Jafar}@imag.fr
[2] Depto de Ingeniería Eléctrica, CINVESTAV-IPN, D.F., México
ammartin@mail.cinvestav.mx
[3] Ciencias de la Computación, CICESE, Ensenada, B.C., México
favela@cicese.mx
[4] Facultad de Ciencias UABC, Ensenada, B.C., México

Abstract. This paper is directly focused on the design of middleware functions to support a distributed cooperative authoring environment on the World Wide Web. Using the advanced storage and access functions of the PIÑAS middleware, co-authors can produce fragmented and replicated documents in a structured, consistent and efficient way. However, despite it provides elaborated, concerted, secure and parameterizable cooperative editing support and mechanisms, this kind of applications requires a suited and efficient inter-application communication service to design and implement flexible, efficient, and adapted group awareness functionalities.
Thus, we developed a proof-of-concept implementation of a centralized version of a Distributed Event Management Service that allows to establish communication between cooperative applications, either in distributed or centralized mode. As an essential component for the development of cooperative environments, this Distributed Event Management Service allowed us to design an Adaptive Group Awareness Engine whose aim is to automatically deduce and adapt co-author's cooperative environments to allow them collaborate closer. Thus, this user associated inference engine captures the application events corresponding to author's actions, and uses its knowledge and rule bases, to detect co-author's complementary or related work, specialists, or beginners, etc. Its final goal is to propose modifications to the author working environments, application interfaces, communication or interaction ways, etc.

Keywords: Web cooperative authoring, distributed event management, DEMS, adaptive group awareness inference engine, AGAIE.

1 Introduction

Collaborative authoring is a complex activity; it needs that requirements from the production, coordination and communication spaces [2] be addressed. Based on this, several studies (e.g. [1]) have identified a set of design requirements that

C.A. Coello Coello et al. (Eds.): MICAI 2002, LNAI 2313, pp. 506–515, 2002.

collaborative authoring applications must support, including: enhanced communications, enhanced collaboration awareness [7] - focused and peripheral, and segmented document and version control. The PIÑAS platform [6] addresses these requirements by providing services for collaborative authoring on the Web. The most important PIÑAS features are 1) Seamless distributed cooperative authoring, 2) Full integration to the existing Web environment, 3) Flexible and reliable distributed cooperative architecture, 4) Document and resource naming based on URLs, 5) Document replication for high availability, 6) Elaborated distributed author management, 7) Automatic updating of replicated information, and 8) Session Management and Group Awareness.

However, other needs can be identified besides these ones considering the following scenario: Authors Domingo and Ana are writing a document, working in two of its fragments. This information is shown by the awareness capabilities of the platform (for instance, Radar View). Nevertheless, the first fragment is an image (currently under modification by Domingo) and the second fragment contains the related explanation or comments (currently under modification by Ana). The resulting state of the document will most probably result in semantic inconsistencies if both authors don't become aware of the impact of their changes.

At the fragmentation and permission level, the inconsistencies will pass without being noticed if both authors have the required permission to modify the requested fragments. However, the changes performed to each fragment will indirectly or directly affect the semantic consistency of the document. We highlight that semantic consistency of concurrent productions also constitute a required feature in a collaborative environment. Thus, we introduce the concept and the need of an adaptive and deductive group awareness function based on the capturing and treatment of the cooperating application events.

Inter-application Communication Mechanisms

Inter-applications communication mechanisms are a requirement of collaborative applications and a challenge for their developers. Several studies have addressed this issue and a generic model can be stated based on them: a producer element generates information that may be of interest to consumers. A dedicated mechanism allows the information being produced to be delivered to consumers. In point-to-point communications, the information is sent directly from the producer to the consumer, and a unique Id for each of them is used as the addresses for the communication (e.g. RPC) [10] and Java's RMI [11]). In multicast communications, the information is sent from a producer to a set of consumers (e.g. CORBA Event Service [8], and Java Message Service (JMS) [12]). Further classification can be achieved by considering message-based and event-based communication models.

We next present the Distributed Event Management Service (DEMS) designed for PIÑAS, its implementation, and then introduce the Adaptive Group Awareness Inference Engine (AGAIE) that we designed on top of the DEMS.

2 Design of a Distributed Event Management Service

In the scope of the PIÑAS platform, designed to support a Web cooperative authoring environment, we propose to develop specific mechanisms based on the definition of a dedicated but extensible Distributed Event Management Service (DEMS). Thus, the goal of this part of the work is to design the model and the structure of DEMS that will be in charge of the transmission, management and delivery of group awareness events among all applications (cooperative or not).

The DEMS receives events generated by the applications, manages them (ordering and storage), and transmits them to subscribed applications. Thus, the applications are independent of event transmission and of event diffusion. Moreover, the problem of distributed event transmission concerns only to DEMS, and so it's application transparent.

To be generic, the PIÑAS platform provides the DEMS that allows it to be used by different kinds of cooperative applications. This way, we plan to support multi-user applications capable of providing direct communications (synchronous or asynchronous) among users. All these applications will be combined to constitute a powerful cooperative environment.

From the conceptual point of view, the DEMS (see Fig. 1) is designed as a component to which applications may 1) subscribe to receive notifications (events), 2) perform special actions to configure both outgoing and incoming event flows, and 3) send typed and parametrized events.

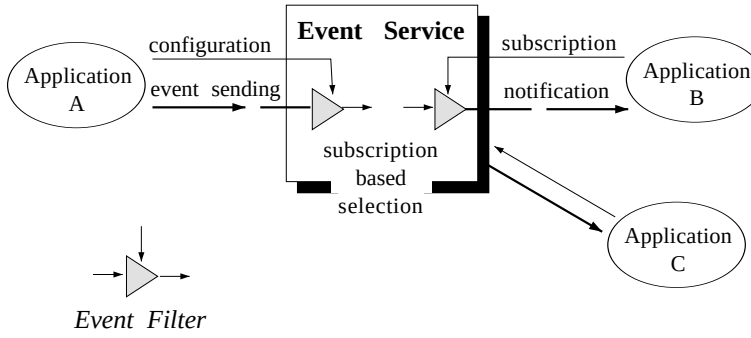

Fig. 1. Schematic Representation of the Distributed Event Management Service

By means of configuration, each application may restrict event diffusion, defining the targeted users and/or applications authorized to receive the events. In the same way, an application may define the kinds of events it is interested in receiving, and only events of these types will be delivered to it.

The event service is a component of the PIÑAS platform, and it transparently runs in both centralized and distributed environments. Thus, it provides support to each application to notify events to other local or remote applications.

2.1 Producer / Consumer Agents

The DEMS has two different sets of clients: 1) *"Producer Agents"* which produce events and transmit them to the service for diffusion, and 2) *"Consumer Agents"* which subscribe to the service in order to receive and consume events.

To be able to send events, a Producer Agent must register itself to the DEMS that returns to it a unique producer Id. Using this Id, the producer agent may perform some configuration actions (see Fig. 1) to extend or restrict the diffusion scope of its events. As an example, some users (i.e. A, B and C) who want to establish a private cooperative session must precise to only diffuse the events to their corresponding colleague's applications.

In a symmetrical way, to capture only certain events, Consumer Agents must provide some rules to filter event categories and/or sources (see Fig. 1).

2.2 Specification of the Distributed Event Management Service

An event is the information unit automatically produced each time an agent performs an action on a shared or private object. To a lesser degree, actions performed on private objects also are interesting to realize elaborated group awareness functions (e.g. expert and novice matching).

Thus, an event is defined as a set of actions executed by an agent on objects[1]:

```
Event = Agent + {Object} + {Action + [Parameters]}
```

Agents

The notion of *Agent* includes the information that allows to identify the nature of the entity that executes the actions on the objects. So, it is not sufficient to identify the active entity (the author), but also the source of these actions (the site and application Ids):

```
Agent = Author + Site + Application
```

This Agent naming principle is based on the PIÑAS author and site naming conventions (see [6]). It maintains and uses the author definition for controlling actions applied to shared resources (documents, images, author entities, etc), and defining sets of roles (e.g. manager, writer, reader, chairman, speaker, reviewer, etc) by which users can act. The role notion is essential to express the social organization of the group work, and constitutes the base for protecting objects from unauthorized actions.

Objects

An event is generated each time an object (usually shared) is accessed. The object's state is characterized by the values of its attributes. A document, a document fragment, an author, and an event are representative examples of

[1] The notation {entity} denotes the possibility to have more than one entity.

objects. The granularity of handled objects may change from one application to another or, more generally, the granularity of executed accesses on the same object varies from one action to another in the same application. For example, a document object is composed of objects of finer granularity such as fragments.

Actions

Actions provide information or represent the user's work: working focus (selection), type of action (cut, paste, copy, open, save), type of object (text, image, annotation), action dynamics (instant, frequency) and their specific attributes (color, bold, plan, importance).

2.3 Configuration and Control of Event Production / Consumption

An important problem to keep in mind is the problem of exporting the variables which compose the user's working environment [9]. To solve this problem, the DEMS provides configurable producer and consumer agents for controlling event production and consumption.

The needs of users and hence the configuration of their work environments evolve regularly in the course of time. These evolution requirements come from their interactions and cooperative processes with other users. For example, in a shared environment, collaborators might change their focus between individual and cooperative work: when collaborators focus on their individual work, they only need general information about the work of others. While they focus on the collaborative task, information must be more specific and more precise. In addition, by means of the shared environment interface, collaborators get familiar with a domain, a task and a group. For example, if a collaborator learns the way other colleagues work, then he can anticipate their actions, and adequate his own, based on the actions they have performed.

Following this principle, a producer agent can authorize, refuse or limit the publication of its events towards certain authorized consumers. Symmetrically, a consumer agent can choose the set of producers.

```
Production_Ctrl = Agent + {Object + Action + {Autorized_Consumer}}
Consumption_Ctrl = Agent + {Object + Action + {Autorized_Producer}}
```

2.4 Event Transmission: A Distributed Service

Cooperative environments typically include group awareness applications (or widgets) that are event producers and/or consumers. For example, cooperative editors like Alliance [3] or AllianceWeb [4] (see Fig. 2) are both producers and consumers of events produced by other co-author applications. Thus, a cooperative editor instance has to be aware of events generated by other co-author applications to reflect and integrate their productions in the author's working environment. In quasi real-time mode, the "radar view" application has to show the different author's working focuses in all working environments.

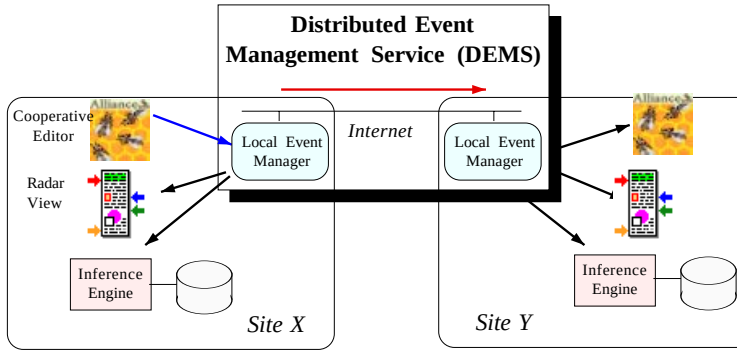

Fig. 2. Event Transmission between Distributed Cooperative Environments

So, considering two users working on two different sites (see Fig. 2) who are respectively using the AllianceWeb cooperative editor and the radar view application: each authoring event has to be diffused both to the local user's radar view and to the remote editor and radar view.

Thus, the DEMS is a distributed PIÑAS service: on each site a local server of this service acts as a concentrator of information (events) on the sender side and as a demultiplexer on the receiver side (site X in Fig. 2). On the sender side, the "Local Event Manager" (LEM) receives events generated by the local producer applications, and if necessary it dispatches them to the local consumers. On the receiver side (site Y in Fig. 2), the LEM instance will determine the applications that are registered as event consumers and distribute the events to them.

Thus, the AGAIE (see section 4) that first appears as a pure consumer of information will also act as an event producer generating/proposing quasi automatic updating of the user cooperative environment (adaptation of the author cooperative editing, proposition of new tools, co-authors, documents, resource and tool appearances).

3 First Implementation of the DEMS

The event management service is a distributed system composed by a set of Local Event Managers (LEM) that are connected together following the peer-to-peer principle. As shown in Fig. 3, each LEM receives the events generated by all local applications and stores them in a dedicated repository. In this event repository, events are classified following their arrival date in the system: it is important to note that no inter-event relation has been investigated and really doesn't appear necessary to establish within the event management system. In a first step, only the production order seems to be relevant.

Thus, the events generated by different applications are mixed within the manager's event repository. For example (see Fig. 3), events $\beta 1$, $\beta 2$ and $\beta 3$ of the cooperative editor β are mixed with the events $\alpha 1$ and $\alpha 2$ of the cooperative editor α.

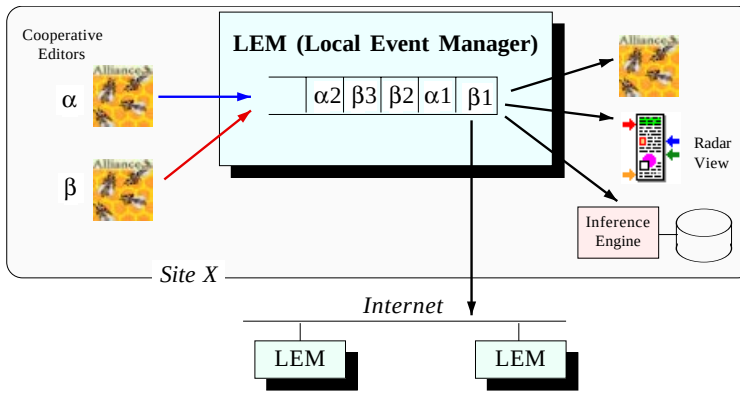

Fig. 3. Functional Principle of a Local Event Management Service

All events are stored in an ordered list implemented as a circular buffer. At the same time the events are stored in the local storage event base, they are also sent to other LEM that take part in the authors cooperative authoring sessions: these remote sites own a local copy of the shared document, and so, they are interested in obtaining all cooperative events related with this document. The LEM service is based on the following principles and constraints:

1. **Event Production** – Each author works with a cooperative authoring application that accesses the shared environment, and spontaneously generates events. The application event layer captures these actions and generates labeled (order ticket) events that are transmitted to the LEM.
2. **Event Consumption** – To obtain events, each client connected to the event manager has to use functions that handle a specific access descriptor. Using and updating specific state variables, this descriptor maintains the current state of event consumption.
3. **Non Blocking Production** – In all cases, this event distribution principle ensures that producers will never be blocked: each event producer must be able to produce new events even when the event storage area is full.
4. **Limited Event Storage Space / Missed Events** – If the limited event storage space becomes full, new elements replace the oldest ones. So, if some consumers (mainly observers) are slower than others, some events may be discarded and so missed.
5. **Dynamic Connection / Disconnection** – The producer and consumer applications may dynamically be connected or disconnected.

4 The Adaptive Group Awareness Inference Engine

One of the specific "group awareness" applications we targeted this study is the design and the implementation of the "Adaptive Group Awareness Inference En-

gine" (AGAIE) whose goal is to define an Adaptive and Deductive Cooperative environment [5].

Running on all co-author's site, the AGAIE is a consumer of all events generated by the co-author's applications (local and remote). Each AGAIE instance makes an "on-the-fly" analysis of the different events it receives. This analysis is performed in association with a knowledge base (rules and inferred data) that is regularly consulted and updated. The final goal of the AGAIE is to deduce information to propose modifications of the corresponding author environments.

The AGAIE functioning principle is organized following the three actions:

1. **The management of a knowledge base** – From the events managed (stored, replicated and/or distributed) by the DEMS, the AGAIE determines and analyzes the corresponding working actions, their sequence and frequencies, their scope of application, the focus and points of interests of the co-authors, and the working and storage author locations.
2. **The deduction of results on the base of pre-designed principles (Rule Base)** – A deductive system is assumed to work from past or known facts to infer or predict new facts. Known facts are the actions already performed by all cooperating authors during and/or out of the cooperative session. New facts are particular information deduced from the action analysis.
3. **The proposition of some new actions (inferred actions)** – that are proposed to be applied on the co-author environment(s).

Among many typical applications of the AGAIE principle (such as the development of the communication between co-authors, coordination support of authors, determination of common author interests, administration of user preferences, etc), the automatic detection of "expert" and "novice" users constitute a realistic and useful goal. Thus, it appears very useful to automatically detect the areas of expertise of some authors in order to help others who had been detected "in trouble" using some equivalent features.

Thus, based on the introduced DEMS, we are developing an AGAIE function able to detect and give marks to the authors in order to automatically determine both "specialists" and "novices/beginners" in the production of (for example) drawings. This Inference Engine is based on:

1. **Determining the focus and complexity of performed actions** – A knowledge base is constituted by the performed authoring actions. All events have to be analyzed to determine their action focus, and if the action is related to a drawing, it is first stored and then its semantics is evaluated. For example, the different actions may create, modify or delete a new graphical item, or modify one of its attributes or to apply some structuring actions (object group creation). During this phase, the point of interests of the author is evaluated as well as the complexity of the performed actions.
2. **Determining the user expertise level** – From the analyzed actions, we can add the temporal dimension: the actions are replaced in the temporal sequence of actions, and an appreciation of the efficiency and the ability of the author are produced. Thus, a user who generated many drawing editing

actions during a quite long time and who only produced a "quite" simple drawing, is classified as a "novice" or a user in trouble. In fact, the aim of this work is quite complex trying to express in rules the definition of an expert: "A drawing specialist is a user able to produce complex and high structured drawings in a quite short time". Of course, the problem is to well express the notion of "complex and high structured drawing", "poor or basic drawing", "short time", etc.

3. **Establishing "expert-beginner" relations** – Then, after determining some specialist and novice users, and regularly updating these evaluations, the AGAIE may propose to a beginner to take a glance, to observe or to analyze a specialist's work, i.e. to be "closer" to the specialist's working environment. Furthermore, the system may propose some special (synchronous or asynchronous) tools to the specialist and the beginner to establish a cooperative session where the specialist will be able to transfer his knowledge and his experience, or at least to give helpful advice to the beginner.

Thus, the AGAIE acts as an event producer whose aim is to propose modifications to cooperative applications. To do that, all event propositions are transmitted to other remote Inference Engine instances or LEMs (see Fig. 3). In this way, we have defined a distributed cooperative system in continuous movement that tries to help the co-authors to be more efficient in their group work.

5 Conclusion and Perspectives

In this work, we presented the design guidelines of a distributed event management service (DEMS) for the PIÑAS platform. This platform, initially designed to support distributed, cooperative and consistent production of documents on the WWW, has been extended to support a fully cooperative environment that includes both cooperative and non-cooperative applications. Currently provides a dedicated service which offers simple but efficient and ad-hoc inter-application notification and communication functions. To achieve this requirement, we designed and implemented a first centralized version of the DEMS that takes into account both the transmission and management of group awareness events among cooperative applications, which are essentially considered as producers and/or consumers of these events.

Also, we presented the example of AGAIE, an adaptive inference engine whose main goal is to capture all produced/exchanged events among cooperative applications. By analyzing these events on the fly using the inference rules of a database and based on the already deduced properties or actions, the AGAIE inference engine proposes actions to adapt co-author working environments in order to make their collaboration more efficient.

The PIÑAS platform is currently being validated by the development of the AllianceWeb cooperative Web browser/editor and a radar view application. These applications along with the AGAIE inference engine and other PIÑAS components, are being developed in parallel, to have them progressing in a concerted way.

Acknowledgements

This work is supported by the ECOS and ANUIES organizations (project M98M01), by CONACyT projects (29729-A and 33067-A), by CNRS-CONACyT projects (9018 / E130-505 and 10395 / E130-706), and by SEP-SESIC and UABC (grant P/PROMEP: UABC-2000-08), with the scholarship UABC-125 provided to Mr Alberto L. Morán.

References

1. R. Baecker, D. Nastos, I. Posner and K. L. Mawby, "The User-Centred Iterative Design of Collaborative Writing Software", *In Proc. of ACM/SIGCHI and IFIP Conference on Human Factors in Computing Systems (INTERCHI'93),* ACM Press, Amsterdam (Netherlands), pp. 399-405, 24-29 April 1993.
2. G. Calvary, J. Coutaz, and J. Nigay, "From Single User Architectural Design to PAC*, a Generic Software Architecture for CSCW", *In Proc. of the Conference on Human Factors in Computer Systems (CHI'97),* ACM Press, Atlanta, Georgia (USA), pp. 242-249, 22-27 March 1997.
3. D. Decouchant, V. Quint and M. Romero Salcedo, "Structured and Distributed Cooperative Editing in a Large Scale Network", *Groupware and Authoring (Chapter 13),* R. Rada, ed., Academic Press, London (Great Britain), pp. 265-295, May 1996.
4. D. Decouchant, A. M. Martínez and E. Martínez, "Documents for Web Cooperative Authoring", *In Proc. CRIWG'99, 5th International Workshop on Groupware,* IEEE Computer Society, Cancún (México), pp. 286-295, 15-18 September 1999.
5. D. Decouchant, A. M. Martínez, "A Cooperative, Deductive, and Self-Adaptive Web Authoring Environment", *In Proc. of the Mexican International Conference on Artificial Intelligence (MICAI'2000),* Lecture Notes in Artificial Intelligence, Springer Verlag, Acapulco (México), pp. 443-457, 11-14 April 2000.
6. D. Decouchant, J. Favela and A. M. Martínez-Enríquez, "PIÑAS: A Middleware for Web Distributed Cooperative Authoring", *In Proc. of the 2001 Symposium on Applications and the Internet (SAINT'2001),* IEEE Computer Society and IPJ Information Processing Society of Japan, San Diego, California (USA), pp. 187-194, 8-12 January 2001.
7. A. L. Morán, J. Favela, A. M. Martínez and D. Decouchant, "Document Presence Notification Services for Collaborative Writing", *In Proc. CRIWG'2001, 7th International Workshop on Groupware,* IEEE Computer Society, Darmstadt (Germany), pp. 125-133, 6-8 September 2001.
8. Object Management Group., *"CORBA Event Service Specification, Version 1.1",* 2001, http://www.omg.org/technology/documents/formal/event_service.htm.
9. D. Salber, J. Coutaz, D. Decouchant and M. Riveill, "De l'observabilité et de l'honnêteté : les cas du contrôle d'accès dans la Communication Homme-Homme Médiatisée", *In Proc of the conference "Interface Homme-Machine" (IHM'95), CEPAD,* pp. 27-34, 1995 (In french).
10. Sun MicroSystems Inc., *"RFC 1050, RPC: Remote Procedure Call Protocol Specification",* 1988, http://www.faqs.org/rfcs/rfc1050.html.
11. Sun MicroSystems Inc., *"Remote Method Invocation Specification Version 1.3.0",* 1999, http://java.sun.com/j2se/1.3/docs/guide/rmi/spec/rmiTOC.html.
12. Sun MicroSystems Inc., *"Java Message Service Specification Version 1.0.2b",* 2001, http://java.sun.com/products/jms/docs.html.