Ariane: An Awareness Mechanism for Shared Databases*

Vaninha Vieira[1], Marco A.S. Mangan[1,2], Cláudia Werner[1], and Marta Mattoso[1]

[1] Computer Science Department, COPPE, Federal University of Rio de Janeiro – Brazil
C.P. 68511, Rio de Janeiro, RJ, Brazil – 21945-970
{vaninha,mangan,werner,marta}@cos.ufrj.br
[2] Faculdade de Informática, PUCRS - Rio Grande do Sul, RS, Brazil

Abstract. Awareness is an essential requirement in collaborative activities. This paper presents Ariane, a generic and reusable awareness infrastructure, independent of a specific application or DBMS. Ariane improves the availability of awareness information to different cooperative applications by monitoring the application persistence mechanism. A prototype of Ariane was developed using the Java Data Objects (JDO) persistence mechanism and aspect-oriented programming techniques, which were employed in order to increase the potential reusability of the solution. A preliminary evaluation of the prototype, applied in an environment for cooperative software development based on components, confirmed that no additional code is necessary to monitor JDO complaint applications. Besides, Ariane proposes a multidimensional data structure for awareness information, the awareness cube. On-line analytical processing tools can be employed to perform queries to retrieve aggregated value from small grained awareness information.

1 Introduction

Awareness in cooperative work means the knowledge and the understanding of things that happen or have happenned in the context of the group which are relevant for the accomplishment of the activities of the participants. The lack of awareness could result in several problems, such as unmotivated people, conflicts, duplicated or inconsistent work [1][2].

Most of the awareness mechanisms proposed in the technical literature develop specific solutions for particular problem domains. These approaches are hard to generalize in different situations, compelling cooperative application designers and developers to recreate awareness solutions in every new application [3].

In cooperative applications, users commonly interact through shared artifacts manipulation [4]. In order to guarantee the artifacts sharing and durability, database systems are regularly used for persistence. The knowledge of the changes performed in the application database helps to clarify the interactions occurred in the group and further improve the group awareness.

This paper presents Ariane, an awareness mechanism based on the monitoring of application databases. The main innovation of Ariane is that it is a generic and flexible mechanism, independent of any specific application or database system. To

* This work was partially funded by CAPES and CNPq agencies.

G.-J. de Vreede et al. (Eds.): CRIWG 2004, LNCS 3198, pp. 92–104, 2004.

achieve flexibility and independence, Ariane was developed using Java Data Objects (JDO) [5], a standard proposal for transparent persistence in Java applications, combined with Aspect Oriented Programming (AOP) techniques [6]. These technologies enable the selective activation or deactivation of the awareness mechanism, according to the application needs. In addition, extensions to the mechanism can be developed with reduced effort. A contribution of Ariane is the proposal of an awareness cube, a multidimensional structure to store awareness information which enables OLAP and data mining tools to be used over past awareness information to analyze group interaction and to discover hidden knowledge about the work of the group.

Ariane is a component of the OdysseyShare project [7][8]. OdysseyShare is a collaborative version of the Odyssey Software Development Environment (Odyssey SDE) [9], which supports component-based software development. OdysseyShare aims to provide a set of tools for group interaction support to be used in Odyssey SDE. The main contribution of Ariane to OdysseyShare is the monitoring of its persistence mechanism, providing collections of awareness information. These collections are explored by awareness widgets of OdysseyShare, which are responsible for data filtering and organization in order to produce appropriate information to different user categories of the OdysseyShare SDE.

This paper is organized as follows: the next section introduces the awareness problem in cooperative applications; Section 3 describes Ariane, its awareness process and architecture; Section 4 presents the Ariane prototype and its use in OdysseyShare; related work is discussed in Section 5; and, finally, Section 6 concludes this paper with some final considerations.

2 Awareness in Cooperative Applications

A cooperative application should produce awareness information about things that happen or have happened in the group context, reporting this information to group members, in order to improve the interaction between members of a group and to enable them to coordinate their own activities. The awareness information can be related to: the group composition (Who are the group participants? What are their skills? Are they available?), the group objectives (What are the activities that should be executed? How will each participant contribute?), and the group activities execution and coordination (What activities have already been finished? What needs to be done? Are the members having problems?). Awareness mechanisms are defined as techniques implemented by a system that aim to support the generation of awareness information through the monitoring of the overall group activities, and the distribution of that information to interested group members.

Through awareness information, participants can coordinate their own activities relating them to activities of other participants, discovering and solving problems such as conflicts, duplicated or inconsistent work. Besides, the group coordinator can benefit from awareness information to identify and resolve high levels of conflicts, premature decisions and lack of participation [10]. If the coordinator had mechanisms that gave him relevant information about events, he probably would find it easier to lead the group to a successful accomplishment [10].

The role a participant plays in the group is important to determine the kind of awareness information he/she is interested in. Ariane considers three main roles: operators, coordinators and analysts. *Operators* are those who execute the group activi-

ties working in a cooperative way through a shared workspace. They are interested in information about occurrences related to their own activity, as for example actions over artifacts they are or were working on. Most awareness mechanisms proposed in the technical literature are designed to help this kind of users, specially when they are working synchronously.

Coordinators are users who are responsible for activities related to make the group work well and focused on their tasks [10]. They need summarized and aggregated information about planned and executed activities to identify situations where their intervention might be necessary, such as when a user has a low level of participation, when there are high level of conflicts or low level of interaction between participants. Borges and Pino [10] have identified these activities and related awareness information to propose awareness tools to support this kind of users.

Analysts are advanced users who are responsible for analyzing the overall work of different groups in an organization in order to discover things about the group that help them making decisions or defining strategies for the group. High level analysis queries might be "Which users are the most participatives?", "Which users have abilities to play the coordinator role?", "How can I improve the productivity of my groups?", "Which users work better with whom?" and so on. To answer these kind of questions is not easy. The technical literature lacks discussions and solutions to support this kind of user in cooperative applications.

The specific objective of Ariane is to support the gathering and distribution of awareness information related to asynchronous interaction for coordinators and analysts. The next section describes the proposed mechanism.

3 The Ariane Awareness Mechanism

The Ariane awareness mechanism was designed to provide awareness information in a flexible and non-intrusive way taking advantage of the application persistence process to collect awareness information. Thus, no change is required in the application code since persistence is a functionality that is implemented in most applications.

Monitoring the application database can provide a great deal of information about the actions performed by the group members. However, some actions can not be monitored. To be monitored the action must generate communication between the application and the database. Thus, actions such as mouse moving or bar scrolling are not captured by Ariane.

Ariane is based on event notification. Events are structured messages containing information needed to promote awareness. There are four event types: **Session Event**, refers to actions of opening and closing a connection to a database, **Transaction Event,** indicates that a begin transaction, commit or rollback action occurred in the database, **Change Event**, reports create, update and delete actions, and **Query Event,** concerns retrieve actions. Change and query events can refer to either the database data or the database schema.

The event structure in Ariane follows the 5W+1H format, which indicates **who** executes **what** action, the date/time **when** the action occurred under an artifact (**where**) for what reason (**why**) and **how** it happened. The **why** and **how** questions are very difficult to answer in an automatic way, because the former implies a knowledge about what the user was thinking when he/she executed the action, and the latter implies a knowledge of the application model. These questions were considered in the

event structure of Ariane, but they still lack appropriate answers. The next subsections will describe the awareness process used by Ariane and the Ariane architecture.

3.1 Awareness Process

The Ariane awareness process (Fig. 1) consists of four phases: event production, distribution, consumption and analysis, and ten activities described in the following.

The **event production** phase takes place with the following activities: (i) the monitoring of the communication between the application and the database to capture actions performed by application users (awareness producers); (ii) the generation of event messages containing the 5W+1H information, which are gathered from the actions captured; and (iii) the storage of the events in the event repository.

In the **event distribution** phase, there is only one activity: (iv) the events are transported to awareness components previously registered as event listeners.

The **event consumption** phase has an activity where (v) visual awareness components prepare and present the awareness information in an appropriate representation to final users (awareness consumers). These awareness consumers might be playing the role of an operator or a coordinator, as defined in the previous section.

The **event analysis** phase aims to prepare the generated events for use in tools based on the On-Line Analytical Process (OLAP) model. To achieve this, the first step is to prepare the awareness information and store it in a special storage structure based on multidimensional modeling techniques called *awareness cube*. To populate the awareness cube the events must be (vi) extracted from the event database , (vii) transformed into the multidimensional format, and (viii) loaded in the awareness cube.

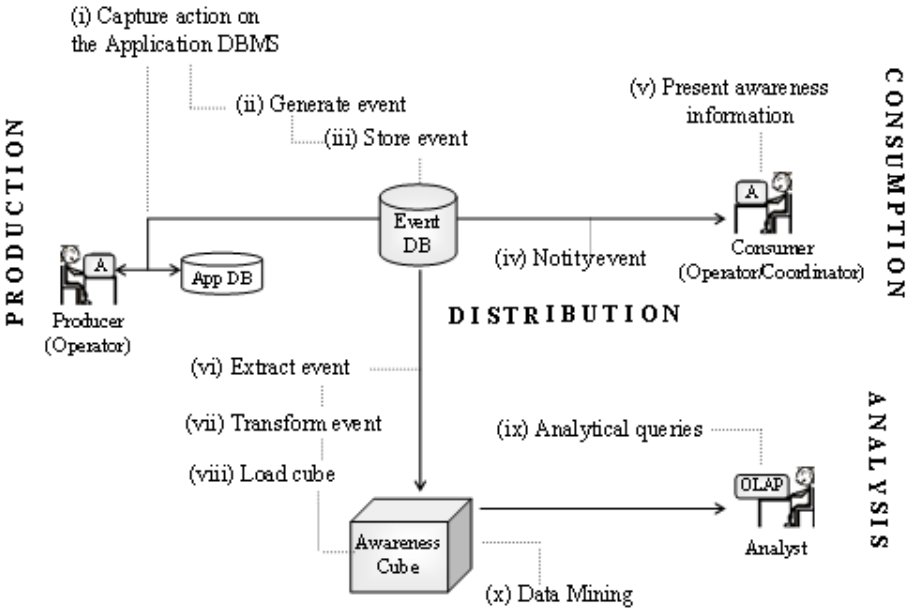

Fig. 1. The Ariane Awareness Process

Using the awareness cube an analyst can: (ix) use any OLAP tool to execute analytical queries, or (x) apply data mining techniques to discover knowledge about the group interaction.

The analysis phase in our awareness process and the proposal of a multidimensional structure is a first step to support the analyst role. As we have said in the previous section, support for analyst users is not stressed in previous awareness mechanisms. The multidimensional model enables flexible visualization of the information, since the user can freely choose and change the axis and rotations for data visualization. Users can choose the granularity they want to visualize the data and the way they want to see the information (pivot table, graphics, etc).

Awareness filters must be applied in every phase to decrease the information overload. In the event production phase, the actions collected are filtered by their type, so that only actions related to persistent artifacts are monitored. Additional filters are not considered in this phase because the focus is to monitor the overall communication between the application and the database. Therefore, the sequence of actions are kept and can be rebuilt afterwards by the same, or another, application. The distribution phase uses event types to filter events that should be delivered. Events are classified by their type and the visual awareness components must register their interest in specific event types, so that they only receive events from the type that they are registered for.

In the consumption phase, filters should be created according to the final user profile and must be implemented by the visual awareness components. Finally, in the analysis phase, filters are provided by the OLAP tools.

3.2 Architecture of Ariane

Fig. 2 illustrates the architecture of Ariane. The components designed and implemented by Ariane are represented by ellipses, and they are the frames named *client* and *awareness server*. Visual awareness components appear to emphasize their role in the awareness solution. However, cooperative application developers must design and implement specific visual components that match the awareness needs of the application users.

The general operation of the architecture occurs as follows: first, *operators* (producers and consumers) interact with instances of a cooperative application, which uses a persistence mechanism to store their artifacts in a database (this flow is represented in Fig. 2 by dotted arrows). A *sensor* is plugged in to the persistence mechanism and listens to the communication between the application and the database, reporting all collected information to the awareness server. In the *awareness server*, the *event handler* component checks the event type, creates an event message and sends the event to all registered components, including the *storage handler* component. The *storage handler* is responsible for storing the event in the *event DB*, reading the events already stored, and answering queries performed by visual awareness components. The *visual awareness components* receive events from the event handler (in a synchronous mode related to the action occurence) and from the storage handler (in an asynchronous mode). They extract the awareness information from the event messages, prepare and present them to the consumer users, considering their roles and profile. Periodically, *ETL (Extract-Transform-Load) processor* component catches

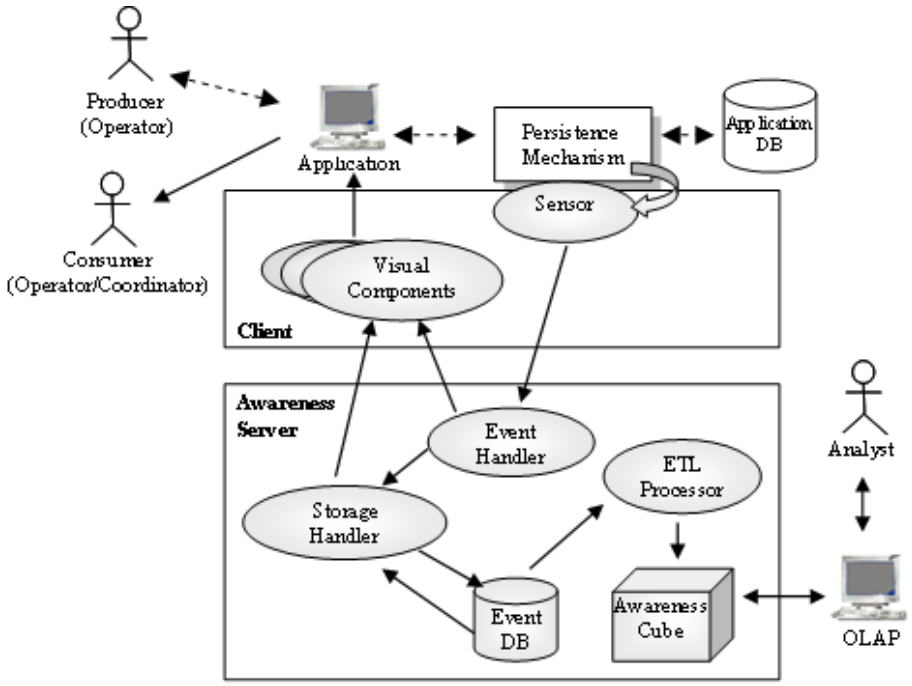

Fig. 2. Architecture of Ariane

events from the *event DB* and loads them in the *awareness cube, where* the analysts can use OLAP tools to execute *ad-hoc* analytical queries.

4 Ariane Prototype Development

In order to validate the feasibility of the Ariane approach, a prototype of the mechanism was developed in the Java platform. Communication between clients and the awareness server uses Remote Method Invocation (RMI) [11]. The next sections discuss details related to the development of the main components of Ariane.

4.1 Sensors

Ariane is proposed as a flexible and generic awareness mechanism, independent of a specific Database Management System (DBMS) or application. The first concern in the development of the prototype was to define where exactly the Sensor should be placed in such a way that it could monitor and collect the actions in a non-intrusive way, requiring no changes in the application and in the DBMS. To accomplish this, we analyzed the communication between a generic application and a DBMS...

Java was our development platform, therefore, we decided to monitor Java applications using a standard interface for persistence: Java Data Objects (JDO) [5], that

specifies a set of interfaces that define transparent persistence. Using JDO, the application is capable of persisting artifacts using any JDO compliant DBMS. Currently, JDO has many implementations (commercial and open source) and is being considered as a standard for persistence in Java desktop applications. Also, is crucial to make sure that the collection of awareness information is done over the object-oriented application data model and not the underlying persistence relational data model. This issue is fundamental in the event consumption phase when the awareness information gathered can easily be understood by the application users, since its semantic is in the same level of the application semantics.

The Sensor must be connected to a JDO-compatible persistence mechanism (JDO implementation) to monitor the events. The second main issue concerning the prototype development was how to implement the Sensor without changing a specific JDO implementation "by hand"? We must consider that better and more robust JDO implementations are commercial and their source codes are not available to be changed.

To solve this issue, Ariane implements the Sensor as an *aspect*. Aspects constitute the programming unit in the software development paradigm called Aspect Oriented Programming (AOP) [6]. Aspects describe and implement application crosscutting concerns, clearly separating them from the application base code. A weaver mechanism merges the aspect code with the application base code so that they are compiled as one single unit. Ariane uses AspectJ [12], that extends the Java language with AOP constructs. AspectJ enables weaving over bytecodes, therefore it is not necessary to access, to know or to understand the JDO implementation source code, since the Sensor, as an aspect, specifies the monitoring code based on JDO standard interfaces . Fig. 3 ilustrates the overall idea of how AOP works in Ariane.

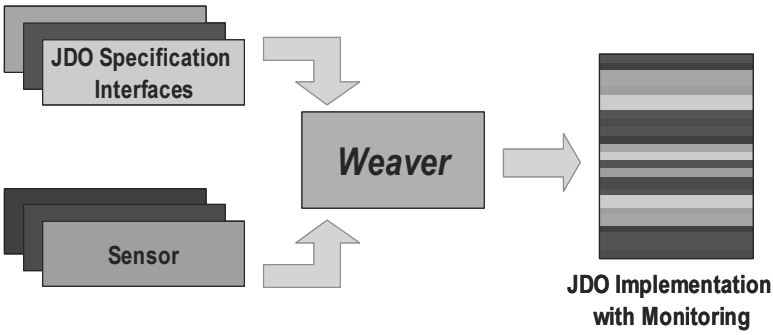

Fig. 3. Weaving of sensor code and JDO Integration interfaces using AOP

One important advantage in using the AOP approach is that it makes possible to implement additional sensors. Thus, different event types could be defined (as, for example, mouse moving or key pressing) or even the monitoring of different persistence mechanisms or the application itself could be implemented is a similar way.

4.2 Event Handler

The event handler receives, from the sensor, information about the action and creates correspondig events according to the 5W+1H format. These events are propagated to

the storage handler and to all visual awareness components that have registered interest in the event type. This propagation was developed following the model used in the Java Abstract Windowing Toolkit (AWT) and Java Beans [13]. The events are classified by their types and each event has a corresponding event class and listener interface. The visual awareness components should implement the listener interface related to their desired event types and should register themselves as interested in the event handler. When a new event is created, the event handler verifies all registered listeners for that event type and sends them a notification message.

4.3 The Awareness Cube

Multidimensional modeling is a discipline that structures data with the purpose of analysis and performance and is a common format in OLAP tools [14]. The modeling of the awareness cube is based on two structures proposed in the technical literature: the CRUD Cube [15] and the DataWebHouse [14]. The awareness cube was modeled using the star scheme, a widely used form of multidimensional modeling. This scheme consists of one central table, named the fact table, which generally contains a huge amount of data and is linked to several peripheral tables, named dimension tables, which qualify the data. In Ariane, the fact table stores the events and the dimension tables store the 5W+1H information, increased with as much information as the Sensor can capture.

A standard ETL processor was used to load the awareness cube with data extracted from the event database using an ETL procedure.

4.4 An Example of Use of the Ariane Prototype

Some tests were conducted with the Ariane prototype using the OdysseyShare SDE. OdysseyShare SDE is developed in Java and can use several persistence mechanisms. JDO compliant or, partially compliant, mechanisms include the JDO Genie [16] and the persistent object manager GOA [17]. GOA is currently the OdysseyShare main persistence mechanism due to its capability of manipulating distributed and mediated databases [18], and XML documents [19].

We choose to monitor the JDO Genie persistence mechanism since it is considered one of the best and more popular JDO implementations available. To connect the Ariane sensor and the JDO Genie, the AspectJ weaver was executed over the main .jar file of the JDO Genie generating a new .jar file, weaved with the Sensor functionality. The new JDO Genie .jar file started to be used by OdysseyShare instead of the old one. No changes had to be done in OdysseyShare in order to include the monitoring functionality. To OdysseyShare users, the monitoring occurs in a transparent way and they proceed using the application as before.

Two widgets were constructed to present the awareness information gathered from monitoring OdysseyShare: (i) *ConnectionReport* (Fig. 4) that shows Session Events, reporting which user has connected to which database, open connection time (open column) and the time they have disconnected (close column), and (ii) *EventMonitor*, a tabular interface that shows all events, and all 5W+1H information gathered, acting as the application log.

Widget ConnectionReport

User	Database	Open	Close
vaninha	jdbc:microsoft:sqlserver://l...	Tue Oct 21 15:09:0...	
vaninha	jdbc:microsoft:sqlserver://l...		Tue Oct 21 15:09:2...
mangan	jdbc:microsoft:sqlserver://l...	Tue Oct 21 17:00:4...	
mangan	jdbc:microsoft:sqlserver://l...		Tue Oct 21 17:10:4...
mangan	jdbc:microsoft:sqlserver://l...	Tue Oct 21 17:13:1...	
mangan	jdbc:microsoft:sqlserver://l...		Tue Oct 21 17:18:4...
mangan	jdbc:microsoft:sqlserver://l...	Tue Oct 21 17:21:0...	
mangan	jdbc:microsoft:sqlserver://l...		Tue Oct 21 17:21:4...
vaninha	jdbc:microsoft:sqlserver://l...	Tue Oct 21 17:26:2...	
vaninha	jdbc:microsoft:sqlserver://l...		Tue Oct 21 17:34:3...

Fig. 4. Visual Awareness Component displaying server connection events

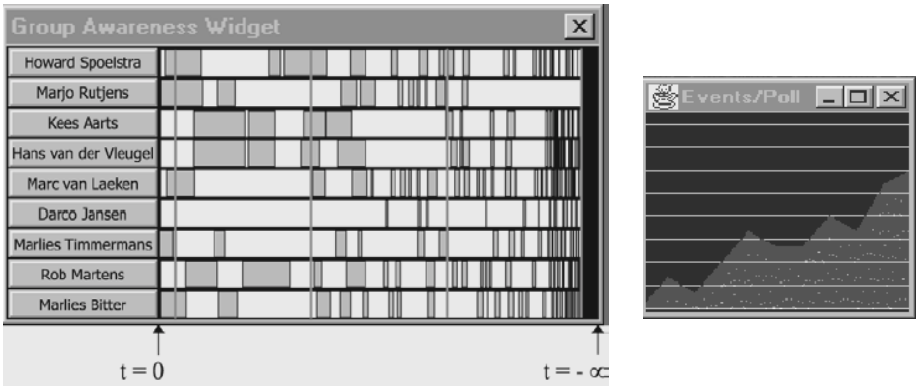

Fig. 5. Visual Awareness Components (a) GAW [20] and (b) Awareness Gauge [21]

The events produced by Ariane can be used as data sources to facilitate the creation of additional visual awareness components that match the needs of the application users. The same information can be visualized in different ways as, for example, using a Group Awareness Widget (GAW) [20], a visual component that shows users connected to a system during a period of time (Fig. 5a), or the Awareness Gauge Events/Poll [21], a component that presents an amount of events occurred in periods of time (Fig. 5b). Therefore, Ariane helps to reduce the effort of visual awareness component developers, since they can just concentrate their efforts in presentation issues.

An additional possibility for event visualization is provided by OLAP tools (Fig. 6). These tools provide functionalities that help analysts to understand patterns and trends in large collections of data. An analyst aware of the group objectives and activities can benefit from this information to evaluate group performance and detect situations where an intervention is needed.

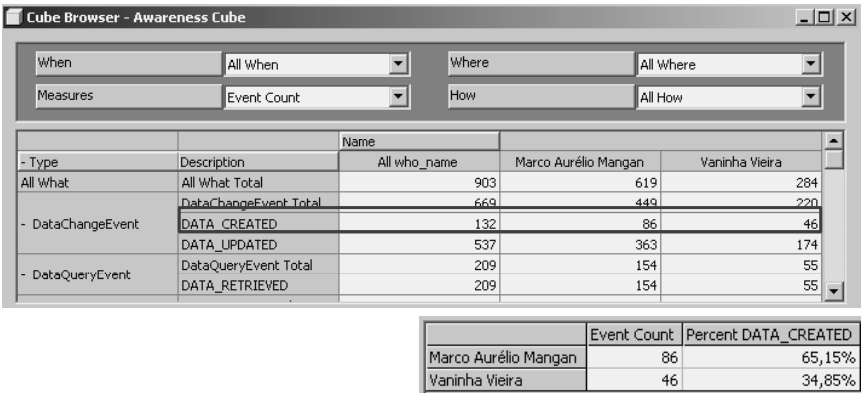

Fig. 6. Awareness Cube – Example of query using an OLAP tool

5 Related Work

There are several awareness mechanisms proposals. Most of them are strongly coupled to a specific cooperative application. Few approaches propose generic and reusable awareness mechanisms, such as SISCO [22], NESSIE [23] and BW [1]. Also, none of the works found reports the use of OLAP to improve awareness in cooperative applications.

SISCO is a cooperation filter for a multi-user generic object-oriented database (OODBMS). Awareness information is gathered by trapping all accesses made to the shared objects from both cooperation-aware and unaware database clients. The filter is responsible only for the cooperation mechanisms, therefore the underlying database is left unaltered. However, the SISCO prototype implementation was done through extensions to a specific OODBMS, the Oggetto. Thus, it is restricted to applications that use that OODBMS. Also, SISCO imposes changes in applications that must connect to the filter instead of the DBMS. Since Ariane monitors a standard persistence mechanism, there are no restrictions related to the DBMS used by the application and no changes in the application are necessary.

NESSIE is an awareness infrastructure for cooperative environments. Key elements of NESSIE are an application independent generic infrastructure, an open and extensibleprotocol including dynamic event types, and a set of sensors and configurable indicators, both for discrete and contextual event notifications. However, the events captured by NESSIE sensors come from changes occurred in a specific NESSIE client or the events are explicitly communicated by the applications using HTTP and CGI scripts. In Ariane, events are generated in an automatic and non-intrusive way, being gathered from the current persistence process of the application.

Finally, BW is a framework that supports asynchronous awareness of past events and has been designed to be used when developing new groupware applications and also to improve existing ones. By being a framework, BW does not implement all functionality needed to promote awareness, demanding extensions to be done in the groupware application in order to use its services. Also, BW forces applications to communicate directly event ocurrencies. Ariane can be used as a plug-in component, which can be connected to or disconnected from the application, without changing its behavior.

6 Conclusion

This paper presents a generic and reusable awareness infrastructure independent of a specific application or DBMS. Ariane improves the availability of awareness information to different cooperative applications by monitoring the application persistence mechanism. This infrastructure can be reused in the construction of many cooperative applications and awareness components.

Ariane monitors application actions performed over its persistent artifacts and consequently over the application database itself. Ariane uses an awareness process based on four tasks: production, distribution, consumption and analysis of events. Events are produced by sensors, coupled to the application persistence mechanism. Event data can be easily consumed and interpreted by different visual awareness components. Events are stored in a database, therefore allowing the building of group memory. The proposal of an awareness OLAP cube, a multidimensional structure to organize and mine the event database, may help work group specialists to analyze group interactions. OLAP tools allow ad-hoc queries, appropriated to decision making and analysis of group interaction.

A prototype of Ariane was developed using the Java Data Objects persistence mechanism and aspect-oriented programming techniques, which were employed in order to increase the reusability potential of the solution. A preliminary evaluation of the prototype applied in OdysseyShare SDE, an environment for cooperative software development based on components confirmed that no additional code is necessary to monitor JDO complaint applications. Ariane helps cooperative application developers in the construction of new awareness solutions, since they can focus on information processing and presentation issues.

The Ariane prototype opens up many many possibilities of research. Currently, we are evaluating three different applications. First, a future work in the CSCW field is to use this approach to create *context-aware applications* [24], using sensors to help identify context elements. Second, in the software engineering field; another possibility would be the development of *Personal Software Process (PSP) applications* [25], monitoring patterns that would reveal current practices used by a software team. Another application in software engineering would be the construction of operation-based configuration management applications, that need to control all the artifact manipulation occurred in a given period of time.

Current work tries to development of sensor implementations for different persistence platforms and applications and is concerned with the organization of large collections of application event data in different application domains. These data will be explored in future analysis.

Ackowledgments

The authors would like to thank CAPES and CNPq for their financial support. The first author also thanks the UFBA for their support.

References

1. Pinheiro M. K., Lima J. V., Borges M. R. S.: A Framework for Awareness Support in Groupware Systems. In: Proc. 7th International Conference on CSCW in Design, Rio de Janeiro, Brasil, (2002), 13-18
2. Sohlenkamp M., Prinz W., Fuchs L.: POLIAwac: Design and Evaluation of an Awareness Enhanced Groupware Client, AI & Society Journal, v. 14, (2000), 31-47
3. Gutwin C., Greenberg S.: A Descriptive Framework of Workspace Awareness for Real-Time Groupware. In: Computer Supported Cooperative Work, v. 11(3-4), Special Issue on Awareness in CSCW, Kluwer Academic Press, (2002), 411-446
4. Preguiça N., Marting J. L., Domingos H., Duarte S.: Data Management Support for Asynchronous Groupware. In: Proc. of the 2000 ACM Conference on Computer-Supported Cooperative Work, Philadelphia, PA, USA, (2000), 68-78
5. Russell C.: Java Data Objects (JDO) Specification - Final Release. In: http://jcp.org/aboutJava/communityprocess/final/jsr012/index.html, Access in 06/2004
6. Kiczales G, Lamping J, Mendhekar A, Maeda C, Lopes C.V., Loingtier J.M., Irwin J: Aspect Oriented Programming. In: Proc. of the European Conference on Object-Oriented Programming, v. 1241 of LNCS, Springer-Verlag, (1997), 220-242
7. Mangan M. A. S., Araújo R. M., Kalinowski M., Borges M. R. S., Werner C. M. L.: Towards the Evaluation of Awareness Information Support Applied to Peer Reviews of Software Engineering Diagrams. In: Proc. of the 7th International Conference on CSCW in Design, Rio de Janeiro, Brasil, (2002), 49-54
8. Werner C. M. L. et al.: OdysseyShare: an Environment for Collaborative Component-based Development. In: IEEE International Conference on Information Reuse and Integration, Las Vegas, USA, (2003), 61-68
9. Braga R. M. M., Werner C. M. L., Mattoso M. L. Q.: Odyssey: a Reuse Environment Based on Domain Models. In: 2nd IEEE Symposium on Application-Specific System and Software Engineering Technology, Richardson, USA, (1999), 50-57
10. Borges M. R. S., Pino J. A.: Awareness Mechanisms for Coordination in Asynchronous CSCW. In: 9th Workshop on Information Techonologies and Systems, Charlotte, North Carolina, (1999), 69-74
11. Sun: Java Remote Method Invocation (RMI), In: http://java.sun.com/products/jdk/rmi/, Access in 06/2004
12. AspectJ: AspectJ Project Home Page. In: http://www.aspectj.org, Access in 06/2004
13. Sun: JavaBeans Specification. In: http://java.sun.com/products/javabeans/docs/spec.html, Access in 06/2004
14. Kimball R., Merz R.: The Data WebHouse Toolkit, New York, USA, John Wiley & Sons, Inc., (2000)
15. Sulaiman A., Souza J. M., Strauch J. C. M.: The Crud Cube. In: Technical Report ES-616/03. COPPE/UFRJ, (2003), http://www.cos.ufrj.br/publicacoes/reltec/es61603.pdf. Access in 06/2004
16. Hemisphere: JDO Genie. In: http://www.hemtech.co.za/jdo/index.html, Access in 06/2004
17. GOA: GOA Home Page. In: http://www.cos.ufrj.br/~goa/, Access in 06/2004
18. Souza R. P., Costa M. N., Braga R. M. M., Mattoso M. L. Q., Werner C. M. L.: Software Components Retrieval Through Mediators and Web Search, Journal of the Brazilian Computer Society, v. 8, n. 2, (2002), 55-63
19. Vieira H., Ruberg G., Mattoso M. L. Q.: Xverter: Querying XML Data with ORDBMS. In: Web Information and Data Management. In: Fifth International Workshop on Web Information and Data Management. ACM Press., New Orleans, USA, (2003), 37-44
20. Kreijns K., Kirschner P. A.: The Social Affordances of Computer Supported Cooperative Learning Environments. In: 31th ASEE/IEEE Frontiers in Education Conference, Reno, NV, (2001), 12-17

21. De Souza C. R. B., Basaveswara S. D., Redmiles D. F.: Using Event Notification Servers to Support Application Awareness. In: IASTED International Conference on Software Engineering and Applications, Cambridge, MA, (2002), 691-697
22. Mariani J. A.: SISCO: Providing a Cooperation Filter for a Shared Information Space. In: Proc. of the International ACMSIGGROUP Conference on Supporting Group Work: The Integration Challenge, Phoenix, Arizona, USA, New York: ACM Press, (1997), 376-384
23. Prinz W.: NESSIE: An Awareness Environment for Cooperative Settings. In: Proc. of the Sixth European Conference on Computer Supported Cooperative Work, Copenhagen, Denmark, (1999), 391-410
24. Dey A. K.: Understanding and Using Context. In: Personal and Ubiquitous Computing Journal, v. 5 (1), (2001), 4-7.
25. Humphrey W. S.: The Personal Software Process (PSP). In: Technical Report CMU/SEI-2000-TR-022, http://www.sei.cmu.edu/publications/documents/00.reports/00tr022.html, Access in 06/2004.