Defining an Adaptable Mobile Transaction Service

Patricia Serrano-Alvarado* **

LSR-IMAG Laboratory
BP 72, 38402 Saint Martin d'Hères, France
Patricia.Serrano-Alvarado@imag.fr

Abstract. Mobile environments are characterized by frequent variations in connection and bandwidth rates as well as by restrained resources on portable devices. This variability complicates data management and in particular, transaction execution. We consider that to deal with environment variability it is necessary to be adaptable. This paper proposes both a Mobile Transaction Service (MTS) and an Adaptable Mobile Transaction model (AMT) which offer environment awareness and transaction execution adaptability. The MTS is a middleware that besides coordinating the execution of mobile transactions, supports mobile environment awareness. The AMT allows the description of different semantical equivalent ways of executing mobile transactions. Thus, depending on the environment mobile transactions will be executed.

Keywords: Databases, mobile computing, mobile transactions, environment awareness, adaptability

1 Introduction and Motivations

Wireless communication technology, personal digital assistances (PDA), handhelds and portable computers provide a base for mobile computing. We consider a mobile computing environment with a network consisting of Fixed and Mobile Hosts (FH, MH). MHs could be of different nature ranging from PDAs to portable computers. While in motion, an MH may retain its network connections through a wireless interface supported by particular FHs which act as base stations (BS) [10]. In the wireless network (WN), the geographical area is divided into cells, each one covered by a BS. The process during which an MH changes from one cell to another is called *hand-off* or *hand-over*. Mobile environments have particular characteristics. The wireless network communication is generally expensive and highly variable in performance and reliability (bandwidth varies a lot and disconnections occurs frequently). Moreover, MHs resources are very limited in processing, memory and disk capacity as well as in battery power.

* Supported b the CONACyT scholarship program of the Mexican government.
** Thesis advisors: Claudia Roncancio and Michel Adiba, LSR-IMAG Laboratory, {Claudia.Roncancio, Michel.Adiba}@imag.fr

A.B. Chaudhri et al. (Eds.): EDBT 2002 Workshops, LNCS 2490, pp. 616–626, 2002.

These characteristics, complicate data management and reduce the quality of service proposed by applications. For instance, wireless network bandwidth may come down during the execution of mobile transactions (e.g. requested by an MH and executed on a FH). This variation may provoke transaction abortion or long waiting times. Consequently, applications must request their transactions again and again or pay the price of uncertain wireless connection time.

For being useful in mobile computing, distributed data management techniques (e.g. queries, replication, caching, transactions, etc.) should be revisited (i.e. adapted, extended) [13]. In our research, we are interested on transactions and how they could *survive* (i.e. adapt) to the mobile environment variability. Several proposals concerning mobile transactions have been introduced [9,12,17, 5,11]. In the analysis made in [14,15] we find out that the majority of these proposals are particular solutions oriented to specific application context. Except for Moflex [11], proposed works do not take into account the importance of the mobile environment variability. Moflex suggests a transaction definition that adapts its execution when hand-off arrives. Dependencies among sub-transactions are also defined with the objective of having different alternatives in case of failures.

In our thesis work we propose a generic Mobile Transaction Service (MTS) that supports transactions and adapts execution to mobile environment variations. To provide adaptability, the MTS is aware of mobile environment characteristics as bandwidth rate, communication cost, MH disconnections, MH available energy, etc. Adapting transaction execution improves communication costs, response time and application availability, briefly, the application's quality of service. Through the MTS, when environment variations affect transaction execution, the application, instead of aborting or keeping the MH client waiting, will have the opportunity of adapting its transactions (i.e. executing transactions otherwise).

In this paper we introduce the MTS and particularly an adaptable and extended mobile transaction definition. The paper is organized as follows. In Section 2, the MTS objectives and its architecture are introduced. In Section 3, an extended mobile transaction is proposed. Finally, after summarizing related works in Section 4, we present our future works and conclusions in 5.

2 The Mobile Transaction Service

Our work is done in the framework of the NODS project (Network Open Database Services) that aims at defining an open, adaptable architecture that can be extended and customized on a per-application basis [2]. Our approach is characterized by a service oriented view of database functionality. All DBMS functions and related tasks are unbundled into services and applications use services as needed. In the design of services, particular attention is paid on adaptability at different levels. Our MTS will cooperate with other services such as replication [4], persistence [8], event [16] and fault tolerance services as well as with traditional DBMSs.

2.1 MTS Objectives

The literature proposes several interpretations of mobile transactions. For us, *a mobile transaction is a transaction where at least one mobile host takes part in its execution.*

The MTS is a generic service which coordinates the execution of mobile transactions. Its principal goals are to support mobile transactions under different application contexts and to adapt the transaction execution to mobile environment variations. The MTS addresses a wide variety of mobile database applications, in particular those where: (1) MHs request transactions to FHs and maybe only transaction results are locally used, (2) there exist execution interaction between an MH and FHs with some kind of *dependencies*, (3) there exist execution interaction among MHs with dependencies, (4) MH autonomy is needed, due to eventually long disconnections.

Clearly, this kind of applications may access several databases that can be located at mobile and fixed hosts. DBMSs could be heterogeneous having certain autonomy. The MTS acts as a mediator between mobile applications and DBMSs, it actually coordinates mobile transaction executions. For DBMS, interaction with the MTS must be nearly transparent. The MTS should be a normal client requesting transaction execution on behalf of mobile applications (see Section 2.2). This capability allows the MTS to be easily integrated into any DBMS.

As we remarked at the beginning of this paper, transaction adaptability to mobile environment variations is our principal objective. For this, the MTS is aware of the mobile environment and their variations. Besides, it uses the Adaptable Mobile Transaction model (AMT) defined in Section 3. This model allows to describe different semantical equivalent ways of executing a mobile transaction. An adaptable mobile transaction contains a set of *component transactions* that can be distributed on MHs and FHs.

In general, applications send transactions to the MTS. Depending on the mobile environment state (see Section 2.3), the MTS decides how transactions will be executed. After that, the MTS distributes transactions to the appropriate DBMS. During execution, the MTS is aware of the mobile environment. If variations affect transaction executions the MTS adapts them executing transactions otherwise.

The MTS provides a disconnection process. It involves transaction adaptability and storage of information related to transaction states and transaction coordination (e.g. logs). In case of planned disconnections, MH's data versions could be stored on FHs (maybe BSs). If possible, MHs specify the disconnection duration and the re-connection place. This information could improve the transaction execution.

2.2 Global Architecture

The MTS is a middleware between mobile applications and DBMSs. It provides mobile transaction execution to MH applications. The MTS has a three-tier

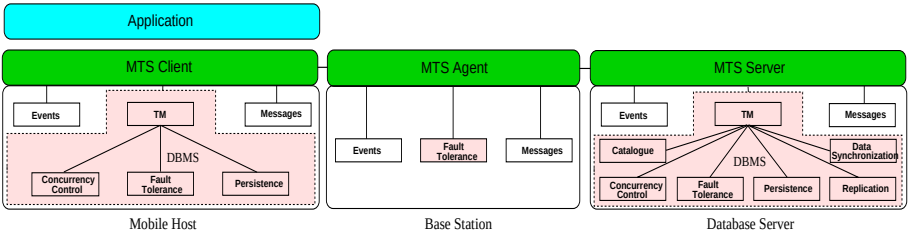

Fig. 1. Global architecture

architecture as client/agent/server (see Figure 1) where the client is on the MH, an agent is on the BS and the server on a FH. As we show in Figure 1, we suppose the existence of database functionalities (e.g. DBMSs or NODS services) on FHs as well as on MHs. The MTS has a set of interfaces to communicate with them. In particular, communication with DBMS is possible by an interface with transaction managers (TM).

The function of the MTS client part is to communicate with mobile applications as well as with local DBMS. Regarding the agent, it is not mandatory to install it on BSs, it could be on a FH able of communicate with a set of MHs. The objective is to have some support on the wired network *close* to MHs to facilitate (1) the interaction with FHs (e.g. storage of communication messages), (2) to allow MH to make backups (e.g. MH's data versions, transactional coordination information), and in particular, (3) to facilitate wireless network awareness (bandwidth rate, communication cost, etc.). The MTS server part interacts with database servers. Besides providing transaction execution, database servers could support process like data synchronization and replication features. In this paper, we make abstraction about replication and synchronization aspects. We consider that MHs contain replicated data in a coherent state.

To support mobile transactions through the MTS, DBMSs do not require significant changes. Applications cooperate with the MTS by designing mobile transactions (as proposed in Section 3) in order to be adaptable to the environment.

2.3 The Mobile Environment

One of the principal characteristics of the MTS is to be aware of mobile environment variations. For this, we characterize the mobile environment with a set of dimensions, given in Table 1. Dimensions may have different values depending on the actual environment state. An *environment descriptor* (ED), is composed by dimensions and their corresponding values reflecting the environment state at a given instant.

Definition 1 *The Environment Descriptor* $ED = \{MEdimension = value\}$ *contains a set of dimensions with their respective values at a given instant.*

For example, ED = {connection-state = *connected*, bandwidth-rate =*high*, communication-cost = *free*, available-battery = *half*, available-cache = *low*, available-persistent-memory = *low*, processing-capacity = *medium*, estimated-connection-time = *00:40:00*}.

Table 1. Mobile environment dimensions

	MEdimension	**Values**	**Unit**
WN	connection-state	*connected, disconnected*	
	bandwidth-rate	*high, medium, low*	kbytes/s
	communication-cost	*free, cheap, expensive*	Euros/time
MH	available-battery	*full, half, low*	hh:mm:ss
	available-cache	*full, half, low*	kbytes
	available-persistent-memory	*full, half, low*	kbytes
	processing-capacity	*high, medium, low*	Mhz/s
	estimated-connection-time	*t*	hh:mm:ss

Values in Table 1 are a first approximation to characterize the mobile environment. In practice, these values may be translated to intervals $[x, y]$. For instance, we can consider that communication-cost[1] is *free* if its value falls in the interval [0,0], *cheap* in [0, 0.2], and *expensive* in [0.3, 1].

3 Adaptable Mobile Transaction Model

To support adaptability, the MTS uses the Adaptable Mobile Transaction model (AMT) proposed in this Section. It allows to define *execution alternatives* semantically equivalent that permit adaptability to environments variations.

3.1 Definition

Execution alternatives describe mobile transactions which are integrated by a set of *component transactions* where:

- a component transaction can be a traditional *flat*, a *distributed* or a *nested* transaction,
- component transactions satisfy the ACID properties.
- the execution grain considered is a component transaction,
- component transactions can be associated to compensating transactions,
- component transactions can be distributed on mobile and fixed host,

At each host, where a component transaction is executed, there exist a transaction manager called *coordinator*. It is responsible of the component transaction execution providing ACID properties.

[1] In this paper, communication-cost is calculated with Euros/time, although, it could be calculated by amount of transmitted information with Euros/kbytes.

According to the informal definition of mobile transaction given in Section 2.1, we classify the *execution models* that a mobile transaction may adopt as:

1. The mobile transaction is initiated by an MH and entirely executed on FHs
2. The mobile transaction execution is distributed between an MH and FHs
3. The mobile transaction execution is distributed among several MHs
4. The mobile transaction is entirely executed on the MH

Mobile transactions with these execution models can be defined with the proposed adaptable mobile transaction.

Execution alternatives will be applied (one at a time) depending on the mobile environment characteristics. For each execution alternative it is defined an environment descriptor which specifies the required mobile environment characteristics for a successful execution. Each execution alternative satisfies semantically the complete execution of AMT. Thus, the successful execution of one of them represents a correct AMT execution. Execution alternatives are defined in a list with the objective of prioritize them. The criterion to position execution alternatives in the list could be, for instance, data consistency. Depending on the replication model (e.g. lazy master replication [9]), it could be considered that data on FHs (master copies) are more consistent than data on MHs (slave copies). To this extent, the transaction programmer may decide that the execution alternatives accessing FHs data will precede the alternatives accessing MH data.

The formal definition of AMT is:

Definition 2 *An adaptable mobile transaction is a triplet* $AMT = (T, CT, ES)$

where:

- $T = \{T_i\}$ is a set of component transactions, $1 \leq i \leq n$.
- $CT = \{CT_j\}$ is a set of compensating transactions, $1 \leq j \leq i$.
- $ES = \langle EA_k \rangle$ (execution strategy) is a list of execution alternatives for AMT, $k \geq 1$, where, EA_k has higher priority than EA_{k+1}.
- $EA_k = (ED_k, EP_k)$, is an execution alternative for AMT. The execution plan EP_k will be executed if the actual mobile environment satisfies the environment descriptor ED_k.
- ED_k is an environment descriptor for a successful execution of EP_k.
- $EP_k = \{T_s, coordinator\}$, is a set that describes a set of component transactions and their corresponding *coordinator*, $T_s \subseteq T$.
  - let $\mathcal{RD}$ be a *relationship dependence* over EP_k, such that, $\forall (T_s, coordinator), (T_t, coordinator) \in EP_k$ exist $(T_s, coordinator) \mathcal{RD} (T_t, coordinator)$

Relationship dependencies can be defined inside execution plans. In the AMT definition, we generalize dependencies with $\mathcal{RD}$. With $\mathcal{RD}$ several types of distributed transactions can be defined. Dependencies can be of different types

(chapter 10 of [7]), we consider execution parallelism (or execution independence) also as a kind of dependence. By space limitations, in this paper we do not describe possible dependencies. As an example, *Begin-on-Commit dependency* ($\mathcal{BCD}$) is a dependence type, where, if we have T_h $\mathcal{BCD}$ T_i the component transaction T_h cannot begin its execution until T_i commits.

The following example shows how an AMT can be described using the proposed formalism.

Example 1 *Suppose that in an archaeological mission at the Lacandona forest in the south of Mexico, archaeologist collect new information about the Mayan civilization. They have laptops and PDAs connected to database servers (located on the fixed network) by a wireless network. MHs can interact with servers to periodically store collected data or to obtain specialized information concerning new data. In this context, we can define an AMT composed of three transactions, where, T_1 is the local writing concerning the collected archaeological object and T_2 is the comparison of this new object with existing ones in the database server. If communication with the server is not possible the comparison will be made locally with the local database (MH), by T_3.*

If we express this example using the AMT model: $AMT = \{\{T_1, T_2, T_3\}, CT_1, \langle(ED_1, \{(T_1, MH_1)\mathcal{RD}(T_2, FH_2)\}), (ED_2, \{T_1, T_3\}, MH_1)\rangle\}$

Component transactions composing AMT are $\{T_1, T_2, T_3\}$. The compensating transaction is CT_1 (read operations do not need compensating transactions). There exist two possible execution alternatives in the execution strategy ES where execution plans are $\{EP_1, EP_2\}$. And the corresponding appropriate state of the mobile environment are $\{ED_1, ED_2\}$, more specifically:

- $ES = \langle EA_1, EA_2\rangle$
- $EP_1 = \{(T_1, MH_1)\ \mathcal{RD}\ (T_2, FH_2)\}$
- $EP_2 = \{T_1, T_3\}, MH_1$
- ED_1 = {connection=*connected*, bandwidth={*high, medium*}, communication-*cost*=*cheap*, available-battery=*half*, available-cache=*half*, available-persistent-memory=*low*, processing-capacity=*medium*, estimated-connection-time= *00:-30:00*}
- ED_2 ={connection={*connected, disconnected*}, available-battery=*half*, available-cache=*half*, available-persistent-memory=*low*, processing-capacity=*medium*}

The MTS will try to apply the execution plan EP_1. For this, it will compare the actual environment state with ED_1. If they do not match, the MTS will try to apply EP_2.

3.2 Adaptability and Transition between Execution Alternatives

The interruption of an execution alternative EA_k will be necessary if changes in the mobile environment contradict one of the environment description dimensions of ED_k. In Example 1, if the execution alternative EA_1 is being executed

and the bandwidth-rate gets into a *low* state, the EA_1 execution will be interrupted. When an EA_k cannot continue, the MTS analyzes if another one could be executed.

The interruption of execution alternatives will provoke the abortion of their component transactions. Compensating transactions could be executed if committed component transactions are aborted. If the interrupted alternative and the next one share component transactions, instead of abort them, they could be taken as part of the next alternative. This will reduce abortions, execution wasting and time executions. Sometimes, compensating transactions could be impossible to define or very expensive. So, there could be committed component transactions that may be just ignored in case of abortion.

Preventing impossibility of communication, the client as well as the agent must be able to take coherent interruption decisions concerning execution alternatives. For instance, if an active execution alternative is using wireless communication and an unexpected disconnection arrives, both the client and the agent must know what is the next step that follows (e.g. next execution alternative to run, waiting for a new connection, etc.). With this, an important part of the internal MTS coordination does not need communication.

To prevent AMT starvation (i.e. the AMT never commits), time-outs can be implemented or the MTS will abort the AMT once it has tried to execute all equivalent strategies without success.

3.3 Environment Awareness with Events

Environment awareness can be implemented with events. Table 2 defines the events type that reflect the variations of the mobile environment (Table 1).

Table 2. Event types for environment awareness

	Event type	**Attached information**	**Notification mode**
WN	*e-connection*	MH id	Immediate push
	e-disconnection	MH id	
	e-hand-off	New BS id	
	e-bandwidth-rate-changes	Bandwidth rate	
	e-communication-cost-changes	Communication cost	
MH	*e-available-battery-changes*	Available battery	Pull before choosing an EA_k
	e-available-cache-changes	Available cache	
	e-available-persistent-memory-changes	Available persistent memory	

The MTS relies on a middleware (e.g. [3]) that provides the environment events (*producer*). The MTS will *subscribe* to events only when needed (*consumer*). The notification modes must be as follows: wireless network events will be notified immediately when they occur by *push-based* techniques, and, MH's events will be notified before choosing an execution alternative by *pull-based* techniques. In push-based notification mechanisms, the producer notifies events explicitly. In pull-based ones, the consumer retrieves events when needed. *e-communication-cost-changes* and *e-bandwidth-rate-changes* events may vary

frequently without affecting the mobile transaction execution. In this case, the MTS will ask the event service to do some filtering to notify events only when changes are significant, i.e. when the corresponding mobile environment dimensions (communication-cost, bandwidth-rate) change of value (e.g. from *medium* to *low*). As we show in Table 2, related information could be attached to events. For example, the *e-communication-cost-changes* event will include the new communication cost.

3.4 ACID Properties

In this Section we position the proposed transaction model against ACID properties (Atomicity, Consistency, Isolation, Durability).

As we said in Section 3.1, *component transactions* satisfy ACID properties. That means that each *coordinator* provides atomic commitment and recovery techniques as well as serializability. Therefore, we consider that AMT with *execution models* of type 1 and 4 (see Section 3.1) guarantee ACID properties.[2]

Nevertheless, providing ACID properties in AMT with execution models of type 2 and 3 becomes very complicated because of the mobility nature of MHs. In these cases the MTS will be responsible of *global atomicity* and *global serializability*. Clearly, we join multidatabase problems because distributed transactions may access different heterogeneous databases. Unfortunately, at present, we have no defined solutions to deal with global atomicity and global serializability in mobile environments. We consider these issues as part of our future work.

4 Related Works

In our transaction model we propose the definition of several semantically equivalent execution alternatives that are similar to *contingency transactions* as used in several advanced transaction models [7], in particular, in DOM [1] and Flex [6]. The DOM transaction model allows *closed nested* and *open nested* transactions. Compensating transactions can be specified for undoing the effects of a committed transaction, contingency transactions can be also specified for being executed if a given transaction fails. In the Flex transaction model, contingency transactions are defined in terms of *functionally equivalent* transactions. A failure order is defined where the execution of a transaction depends on the failure of another one. This model allows the specification of dependencies on subtransactions. They can be of internal or external type. Internal dependencies define the execution order of subtransactions (i.e. failure-dependencies, success-dependencies) and external dependencies define the subtransaction execution dependency on events that do not belong to the transaction (i.e. start and end times). Both, DOM and Flex transactions are not defined to deal with mobile

[2] We consider local data as replicas, thus, in the execution model 4, after disconnection a synchronization process with FHs must be executed. We think that this is more a replication problem than a transaction problem. Clustering [12] and Two-tier replication [9] deal with this issue.

environments. The Moflex transaction [11] is an extension of the Flex model adapted to mobile environments. Moflex allows the definition of *location dependent* subtransactions and defines *hand-off rules* that determine the execution policies for subtransactions when hand-off occurs. Some of the ideas of the proposed adaptable mobile transaction (Section 3) were inspired from Moflex. In our work we deal not only with hand-off but with a set of mobile environment dimensions (connection-state, bandwidth-rate, communication-cost, available-battery, etc.). Further, Moflex was designed to support only the first of the *execution models* introduced in Section 3.1 whereas the MTS supports the four.

As we mentioned in the introduction, there exist several proposals concerning mobile transactions, Two-tier replication [9], Pro-motion [17], Kangaroo [5], etc. Here, we cannot give more details about them, nevertheless, in [14,15] we made a more detailed analysis of these models. Note that adaptability issues are not considered in these proposals.

5 Conclusions and Future Works

We presented a Mobile Transaction Service (MTS) as a middleware that interact with mobile applications and DBMSs. Besides supporting transactions on mobile environments, the MTS provides adaptability with mobile environment awareness. The contributions of our research work are:

- Adaptability of mobile transaction execution based on mobile environment awareness.
- Proposition of an adaptable mobile transaction model which:
  - allows to describe the mobile environment characteristics necessary to execute mobile transactions,
  - allows the description of semantically equivalent alternatives anticipating eventual environment changes,
  - supports different mobile transaction *execution models*.

The MTS is in its design phase. As future works we consider the specification of MTS's interfaces for interaction with *transaction managers*, the *events service* and the *messages service*. Coordination structures with information about transaction state and mobility must be defined. It is also necessary to describe how exactly the disconnection and hand-off process are. Further, message types and communication protocols (among client/agent/server) must be established. Finally, a prototype that will validate the proposed MTS will be implemented.

Besides, we want to analyze the cost of changing from one execution alternative to another and the coherence of this transition.

A very important issue that we will analyze is the correctness criterion. As the MTS can access several heterogeneous databases, correctness could be analyzed like in multidatabase systems taking into account MHs participation.

Acknowledgments. I wish to express my gratitude to my thesis advisors Claudia Roncancio and Michel Adiba for their interesting remarks and helpful discussions.

References

1. A. Buchmann, M. Tamer Ozsu, M. Hornick, D. Georgakopulos, and F. A. Manola. A Transaction Model for Active Distributed Object Systems. In *Database Transaction Models for Advanced Applications*, 1992.
2. C. Collet. The NODS project : Networked Open Database Services. *ECOOP*, 2000.
3. C. Collet, G. Vargas-Solar, and H. Grazziotin-Ribeiro. Open Active Services for Data-Intensive Distributed Applications. In *IDEAS*, Yokahama-Japan, 2000.
4. S. Drapeau, C. L. Roncancio, and P. D chamboux. RS2.7: an Adaptable Replication Framework. submitted paper, 2002.
5. M. H. Dunham and A. Helal. A Mobile Transaction Model that Captures Both the Data and the Movement Behavior. *ACM/Baltzer Journal on special topics in mobile networks and applications*, 2:149–162, 1997.
6. A. Elmagarmid, Y. Leu, and M. Rusinkiewics. A Multidatabase Transaction Model for INTERBASE. In *International Conference on VLDB*, August 1990.
7. A. K. Elmagarmid. *Database Transaction Models for Advanced Applications*. Morgan Kaufmann Publishers, 1992.
8. L. Garc a-Ba uelos. An Adaptable Infrastructure for Customized Persistent Object Management. In *EDBT PhD. Workshop*, Prague, March 2002.
9. J. N. Gray, P. Helland, P.O'Neil, and D. Shasha. The Dangers of Replication and a Solution. In *ACM SIGMOD Conference on Management of Data*, pages 173–182, Canada, 1996.
10. T. Imielinski and B. R. Badrinath. Wireless Mobile Computing: Challenges in Data Management. Communications of the ACM, 1994. 37(10):19-28.
11. K. Ku and Y. Kim. Moflex Transaction Model for Mobile Heterogeneous Multidatabase Systems. In *10th International Workshop on Research Issues in Data Engineering*. IEEE, 1998.
12. E. Pitoura and B. Bhargava. Data Consistency in Intermittently Connected Distributed Systems. In *Transactions on Knowledge and Data Engineering*, November 1999.
13. E. Pitoura and G. Samaras. *Data Management for Mobile Computing*. Kluwer Academic Publishers, 1998.
14. P. Serrano, C. Roncancio, and M. Adiba. Analyzing Mobile Transactions Support for DBMS. In *International Workshop on Mobility in Databases and Distributed Systems in DEXA*, Munich, Germany, September 2001.
15. P. Serrano, C. Roncancio, and M. Adiba. Mobile Transaction Supports for DBMS. In *17i mes Journ es Base de Donn es Avanc es*, Maroc, October 2001.
16. G. Vargas-Solar and C. Collet. A Flexible Event Service for Database Cooperating Components. Technical Report RR 1031-I-LSR-13, LSR-IMAG, France, 2000.
17. G. D. Walborn and P. K. Chrysanthis. Transaction Processing in PRO-MOTION. In *14th ACM Annual Symposium on Applied Computing*, San Antonio Tx, 1999.