How to Manage Spatio-temporal Events in Relational Databases

Dolores Cuadra
Computer Science Department
Carlos III University of Madrid, Avd. Universidad, 30, 28911 Leganés, Madrid (Spain)
Email:dcuadra@inf.uc3m.es

Javier Calle
Computer Science Department
Carlos III University of Madrid, Avd. Universidad, 30, 28911 Leganés, Madrid (Spain)
Email:fcalle@inf.uc3m.es

Jesica Rivero
Computer Science Department
Carlos III University of Madrid, Avd. Universidad, 30, 28911 Leganés, Madrid (Spain)
Email: jrivero@inf.uc3m.es

This paper discusses the need for event-condition-action (ECA) rules in spatio-temporal databases and outlines a generic scenario for them. A proposal for the implementation of ECA rules is then presented in which several implementation alternatives for different spatio-temporal rule types are considered. Through a series of experiments, these implementation alternatives are evaluated to determine which are most appropriate in each case. The article closes with a discussion of practical uses of these alternatives in earlier studies and a section of concluding remarks.

Keywords: Spatio-temporal events, ECA rules, Spatio-temporal Database, Location-Based services.

ACM Classifications: H.2.8, H.3.4

1. FRAMEWORK AND MOTIVATION

In many of today's business organizations, the handling of spatio-temporal information is crucial, particularly in the domain of location-based services. Since a unified treatment of both spatio-temporal and descriptive data is thus highly desirable, the use of spatio-temporal databases is very attractive. Location-based service applications such as traffic and security management systems or electronic tour guide technology are classic examples of areas where reactive system behaviour is needed for user notification. Sent as a signal from a GPS, wireless or RFID device, the location of an object is interpreted by an application (according to domain-related services or business rules) which, under certain conditions, may then send a message to the users. In the described scenario, particular components may be observed such as emitter and receptor devices, a middleware application between the devices and server, a user application to receive messages from the server and a location data management system in the server. This paper considers a scenario in which

Manuscript received: 9 February 2011
Communicating Editor: Clement Leung

object location updates are periodically sent, the future location of the object is unknown and its motion is represented as two-dimensional Cartesian coordinates (x,y) at t (timestamp).

The focus of this paper is moving object information stored in an object-relational database. Location signal management is handled by a physical layer (i.e., middleware) which introduces object location information into the database in a uniform manner, independent of the emitter and receptor devices. Device features, therefore, are most pertinent to this physical layer. This paper considers the implementation of reactive behaviour following the occurrence of a spatio-temporal event from a practical perspective. Taking into account the support that DBMSs offer for the implementation of system behaviour this paper shows which alternative out of those presented is most appropriate for the implementation of reactive behaviour for events of a spatial, temporal or spatio-temporal nature. The article analyses two general ways to tackle this problem, namely, the use of procedures which are internal to the database (internal control) and the use of procedures external to the database (external control), the former taking advantage of the two well-known SQL methods of jobs and triggers. The primary motivation of this paper is to share practical experiences gained through certain projects aimed at providing location-based services in the tourism and academic sectors. These experiences are presented in the practicability study of different methods for implementing reactive behaviour in these types of applications.

2. BACKGROUND

The study of active databases represents a mature, yet constantly-evolving field (Norman and Diaz, 1999) in which new applications in different domains are continuously being developed (e.g., distributed active databases Heimrich and Specht (2009), real-time active databases Qiao *et al* (2007), and embedded active databases Kaiyin *et al* (2009)). In relational databases, some extensions have been defined to specify event-condition-action (ECA) rules supported by specific statements in SQL3 (Eisenberg *et al*, 2003). Among the commercial DBMSs that are most relevant to this study, many give support to triggers as a subset of ECA rules. In general, primitive events have been related to data manipulation (insert, update, and delete), although certain authors such as Chakravarthy *et al* (1994) discuss composite events for active databases. In addition, some metrics are defined to measure the performance (e.g. see Díaz *et al*, 2001).

From a practical perspective, certain problems have been detected in the implementation of active rules in object-relational databases: ECA rules have a largely hidden execution plan; to avoid cycles, DBMSs do not provide all the capabilities for the execution of ECA rules; and, as a result, designers do not have complete confidence in their use (Al-Jumaily *et al*, 2008). The definition of ECA rules, however, is intended as supplementary to the relational model in order to represent semantics not reflected by the latter (Al-Jumaily *et al*, 2002). The definition of ECA rules can check integrity constraints and, in general, can represent behaviour in object-relational databases.

In spatio-temporal databases, the representation of reactive behaviour is essential (Koubarakis *et al*, 2003), especially in moving objects databases (MOD), whose principal features are based more on the motion of an object than on its shape. In one of the more important papers on MODs (Güting *et al*, 2000), a theoretical foundation is developed describing moving objects as defining abstract data types and in which related operations are also described. This paper deals with the implementation of active rules in this type of database. Due to their dependence on geographical information provided by mobile devices, location-based services represent a major area for spatio-temporal data management. In general, MODs are used for the modeling of event-based and context-aware notification systems in mobile environments. In order to provide a mechanism to support reactive behaviour in MODs, the use of ECA rules could present a viable solution. That said, the traditional semantics associated with ECA rules would require adaptation since, while according to traditional semantics a rule is fired when a condition is met by an event, when a spatio-temporal condition is met, the rule could be fired several times (Chen *et al*, 2003). Abraham and Sojan Lal (2008) present a proposal to reduce storage of moving objects in relational databases, thus improving the identification of "sensitive areas" (i.e., locations of special interest) in road networks. To do this, spatio-temporal dimensions are reduced in order to store object information and location history in a binary code and a trigger is used to indicate when a moving object enters or crosses a pre-defined area. Therefore, this approach only takes into account spatial events, dealing with temporal dimension after the spatial events. In Trajcevski *et al* (2005), a new paradigm, $(ECA)^2$, is presented to express active behaviour in MODs. The article defines dynamic predicates which are checked by the updating of user location, that is, with the update operation in the MOD database. A trigger is defined to capture the spatial events, and as usual it can perform any DB action, including self-modifying. While database triggers are used, this approach defines an external module, called the Intelligent Events Manager, to filter and fire triggers. These tasks process temporal conditions, but do not capture temporal events (rules are checked when database events are triggered). Conversely, in Keeney *et al* (2010) temporal events are captured and processed. The interval operations are interpreted through an external module before invoking stored procedures for running the consequences of the rules.

Summarizing, there is a good deal of work involved in developing external processing of spatio-temporal ECA rules, aimed at: decomposition into atomic rules, checking rule consistency, prioritizing atomic rules execution with lazy evaluation criteria in order to increase efficiency, using buffers for the results of the atomic rules execution (which saves on cost because of repeated running of the same rule), etc. However, there are few studies dealing either with embedding the spatio-temporal events capture and processing into the DBMS or with comparing the implementation alternatives for active rules with spatio-temporal conditions in a DBMS.

This paper explores the DBMS mechanisms supporting the implementation of joint spatio-temporal rules, and then compares their performance with one another and with the frequently used external control. The next section presents the framework as a general scenario where the experimental will take place. Section 4 explains the proposed implementations whose comparative evaluation is presented in Section 5. Finally, Section 6 briefly describes a practical example within a research project.

3. FRAMEWORK: THE *STOR* MODEL

The framework for this proposal is based on the spatio-temporal model, STOR (Bertino *et al*, 2003). This model combines OpenGIS specifications with features of SQL3 to extend the object-relational model and provide the necessary semantics for the representation of spatio-temporal objects and data. To do this, new user-data types were defined and metadata was introduced to model the spatio-temporal granularity and coarse function, in order to convert the spatio-temporal granularity based on Bertolotto (1998).

The generic scenario design is illustrated in the UML class diagram below (Figure 1). In the diagram, classes correspond to spatio-temporal objects and are sorted according to topology or position evolution. For the first object type, spatio-temporal granularity represents geometric variations through time, while, for the second type, its evolution (i.e., its position/situation through time) is represented. Each class may be defined by a different spatial and temporal granularity and geometry. These properties add spatio-temporal semantics to the UML class diagram. The diagram describes a scenario in which a user looks for places, objects or persons of interest within an area

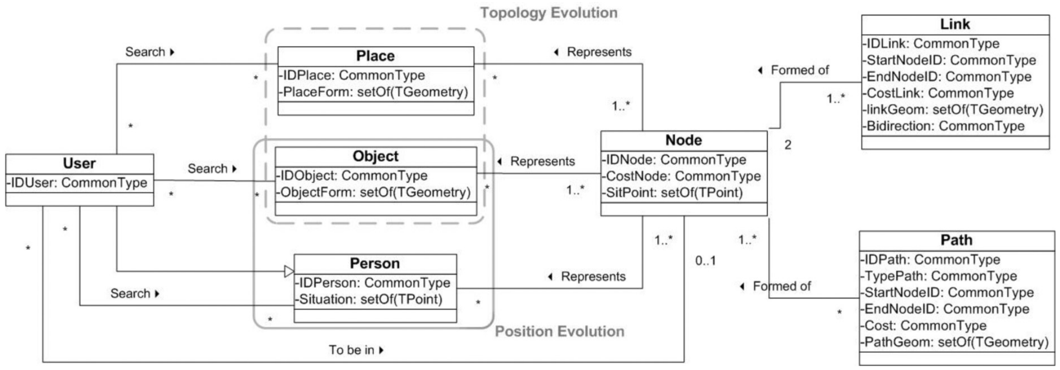

Figure 1: Design to support context-aware applications

referenced in a spatial system (e.g., the metric system) as common material aspects of every domain. The diagram also includes two outlying sections. The first such section (on the right side of Figure 1) is represented by a net graph composed of nodes and links reflecting spatial context (environment). The class, “Links”, describes the practicable way from one particular node to another while the class, “Path”, only observes ways which have already been covered.

The second outlying section (the left side of Figure 1) of the UML diagram consists of elements distributed along the net graph. Nodes may represent interesting elements in the net graph and the net topology evolution depends on the evolution of the “Object” and “Place” classes. Additionally, “Person” and “Object” classes pertain to the second type of spatio-temporal objects (i.e., moving objects). These elements cover the net graph and they are characterized by a position and valid timestamp. The class, “User”, is a specialization of the “Person” class and represents the positional evolution of the current user (i.e., the person interacting with the system). A constraint common to every class is that the position is always valid with respect to the space defined.

Both the environment and the set of moving users are stored in a spatio-temporal database. The architecture of such context-sensitive is shown in Figure 2. Regardless of the tracking device, the middleware receives the signal trace and consequently operates on the DBMS. Those operations will entail (or not) the proper consequences, according to the semantics already defined in spatio-temporal rules (and stored in the rules tables).

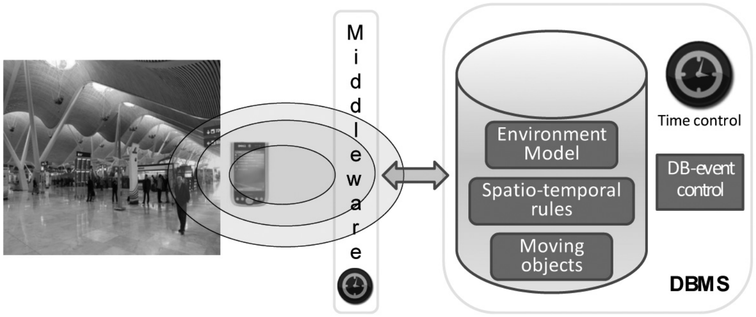

Figure 2: Context Architecture

A spatio-temporal rule can be defined in the following way:

A spatio-temporal atomic rule is a tuple *r* (*ruleID*, *event_type*, *condition*, *action*, *valid-time of the rule*, *priority*), where *ruleID* is the rule identification, *event_type* is a temporal and/or a modification database operation event, *condition* is a temporal and/or spatial expression, *action* is the procedure to be executed, *valid-time* indicates the expiration of the rule (no expiration by default), and *priority* helps to determine the execution order (random if not specified). The set of rules defined for a specific domain are stored in the spatio-temporal rules definition table (hereinafter, the "rules definition table"). An example of atomic spatial rule is shown next:

```
r1: (001, 'spatial', 'SDO_INSIDE(PlaceForm, user_location)',
    'Send_Message(''You are in the namePlace'')', NULL, NULL)
```

This rule would check when a user is within a defined place in the environment model and a message would be sent to him about the name place that he is in.

4. DESIGN OF THE SPATIO-TEMPORAL RULES

The core proposal of this paper is the implementation of active rules that check the validity of specific conditional expressions in stored, spatio-temporal data and, if those conditions are met, fire to execute specific actions. The spatio-temporal information is dynamic, as was the case in the previous example of an object moving through a defined space in real time. The challenge for this proposal is to determine where and how these conditional expressions should be inspected.

Elements receiving a spatio-temporal characterization in the database are subject to a granularity. Given that both space and time are continuous, it is not possible to have them always perfectly updated, but rather only at discrete moments. The information, therefore, must be discretized with both spatial and temporal updates performed according to some concrete procedure. For efficiency reasons, spatial information is not updated unless a significant change is detected, that is, a change of state given a particular granularity.

Additionally, and since real time flow is already known by the server, updating changes of only the temporal information of an element is useless. As a result, temporal data is usually updated with other state information (i.e., spatial data) and is not subject to updating so long as the other state information remains constant.

Consequently, while the meeting of spatial conditions may be checked in the changes recorded in the database, the meeting of temporal conditions must be inspected independently given that temporal events occur even without state updates. Resources for the implementation of these processes may be internal ("internal control") or external ("external control") to the DBMS. Nevertheless, as avoiding a bottleneck in the external control requires job distribution across a cluster of servers, information on active rules should thus be held within the database. External control suppliers will frequently query the active rules stored in order to ensure the validity of the rules set.

Active rules, in general, are subject to a characterization regarding the number of times they should be run (e.g., once or more than once). For example, a rule activating a warning to the user about an area he or she is entering should be run only once, but a warning that he or she is about to leave an area of special interest might be required to be run several times. Apart from that, atomic active rules may also be classified according to the spatial, temporal or spatio-temporal nature of the events according to which the rules are triggered, since event nature may directly affect rule-implementation. In keeping with this last observation, three different rule classifications are considered in the sub-sections below.

4.1 Active Rules with Spatial Conditions

Changes in the spatial state of an object are detected by external processes (e.g., middleware connecting users of physical devices and the database) and sent to the DBMS as an insertion, updating or deleting operation associated to this object.

Thus, condition queries on this information may be processed in two different ways: (1) at the time of data insertion and updating through classic table triggering mechanisms and (2) by the middleware to check if any current active rule conditions have been met when sending the information to the DBMS.

All changes in the spatial information (i.e., positional or morphological changes) of an object are recorded in a table through an update of the latter.

The first mechanism defines a trigger for this table which, in turn, identifies the object for which the change is recorded. Thus, it is not advisable to define a trigger for each rule, but rather for each table containing object data to be checked. This trigger (hereinafter, the "checker trigger") is based on a table set to record the programmed rules. The conditional expression may comprehend any spatial query that the DBMS is able to perform, and the action any self-contained SQL statement or block. When the checker trigger runs, it retrieves the rules relevant to the changed object, tests the rules conditions and, if the condition is met, schedules the immediate execution of the corresponding action.

In the rules definition table, it is useful to include a column indicating whether a rule is enabled or not, so that any rule may be deactivated for a specific period of time if required (*valid_time* of the rule). Finally, given that the action executed according to a specific rule may actually be the deactivation or deletion of another rule, it is also recommendable to include in the table a column establishing a prioritized rule execution sequence (or offering some specific information regarding rule priority, *priority* attribute). In this way, the checker trigger may execute (and perhaps change) rules in the proper order. A similar approach may be observed in the Oracle Database 11G (2009).

The mechanism for the second solution – that involving external control – is quite similar. Given that there could be several middleware servers sharing the same rules, such rules should also be stored within the database. The main difference between internal and external control, then, is that in the latter, middleware is responsible for both event-capture and action-execution.

4.2 Active Rules with Temporal Conditions

Real time flow is discretized into points separated by a constant period of time (i.e., granularity). Whenever one of these discrete points in time is reached, a process is begun to check every active rule with temporal conditions (*event_type* is temporal). Granularity should be coarser than average process time, since if the testing process time were greater than the granule, execution points would have to be skipped (i.e., testing would not be executed at each point, but rather at every two, three or n points).

As was the case with active rules with spatial conditions, two procedures exist for the implementation of rules with temporal conditions: (1) the use of DBMS resources (such as the Oracle Database 11G (2007)) for running periodic checks and (2) the employment of a timer in the middleware to initiate the checking process. As the latter of the two is analogous to the second procedure discussed in the previous sub-section, no further details will be offered here.

The use of the Oracle Job Manager in the checking of active rules opens up several interesting possibilities. Firstly, the Job Manager can be employed for the direct programming of individual rules with solely temporal conditions. The Oracle Database 11G (2007) is then instructed to execute the corresponding action of a particular job at a given moment in time (i.e., encoding the temporal condition). Where appropriate given a specific condition, the Scheduler also enables the periodic repetition of the executed action. For temporal multi-segments, the executed action may also comprehend the redefinition of its corresponding job for future (albeit, modified) execution. Yet, while this solution happens to cover the most common of temporal conditions for events, it should be pointed out that its use is nevertheless restricted to more general and complex temporal conditions.

Additionally, the focus on triggers from the earlier sub-section may be taken up again in the present situation, facilitating a similar rules definition table and a frequent job from the Job Manager to inspect the table (hereinafter, the "checker job"). One potential drawback of this solution, however, is that an average delay of half the checker job's granularity will be introduced into the process. That said, the Oracle Scheduler enables granularity of a second, which should be fine enough for most requirements. Another potential difficulty with this approach is that large workloads may overflow the checker job's processing capabilities, leading to delays in rules inspection and, potentially, to an eventual failure to execute corresponding actions at the right time. In order to minimize this weakness, it is recommended that conditions be ordered chronologically. The solution is helpful in that it saves process time, since the failure to meet one condition will result in the failure to meet conditions later in the chronological ordering. Finally, the solution is relatively scalable: if effectiveness is in danger due to a high workload, several checker jobs can be run simultaneously – with a corresponding rules definition table for each and dividing the work evenly between them – either by creating a job for each object or group of objects, or simply by assigning new rules to jobs with a less voluminous workload.

4.3 Active Rules with Spatio-temporal (Mixed) Conditions

A conditional expression may include several clauses which, when met, result in the performance of a single action. If the clauses are of the same nature (i.e., all spatial or all temporal) they may attach to a single rule; however, conditional expressions including clauses of both spatial and temporal natures require additional consideration here. If the conditional expression is the result of a disjoint (i.e., using the operator "or"), a proposed implementation must employ several active rules (i.e., one for each clause of the condition) interrelated in such a way that if any of the rules is cancelled out, the rest will be removed. If, however, the conditional expression is the result of a conjunction (i.e., using the operator "and"), the appropriateness of the implementation of a rule is not therefore immediately clear and requires further study according to the following methods available: (1) by means of interrelated triggers and jobs, (2) through the sole application of jobs and (3) with the use of checks and timers in the middleware.

It is important to note that triggers alone are not sufficient in the case of such mixed, spatio-temporal conditions, since cases are possible where the spatial, but not the temporal condition is satisfied upon triggering. Additionally, if no further spatial change is recorded, checking would be suspended, regardless of whether the temporal clause of the condition was met. While the alternative procedure of setting a job for each individual rule might, at first, seem to completely solve the problem, in such a case the spatial condition check would be performed as the first step of the action and, were the condition not met, the job would have to redefine itself (i.e., by altering its definition or by creating a cloned job for later testing). It should be noted that such a procedure could result in an all-too-frequent pattern of job execution and alteration for each rule. In summary, none of the solutions considered here seem very practical.

As an alternative to these procedures, an interrelated trigger and job solution could be applied in the following manner:

1. A rule for the checker trigger is set (in its own rules definition table) with the spatial condition.
2. When the trigger reviews the rule, three results are possible:
    - The spatial clause of the condition is not met, then no action is executed and the rule will be reviewed later;
    - Both conditions, spatial and temporal, are met, then the action is executed;
    - The spatial clause of the condition is met, but the temporal clause is not. If such is the case, the rule's validity must then be checked to ensure that the rule condition can be met in the future. If it cannot be, the rule is deleted. If the rule is still valid, then a job will be set with the temporal clause of the condition. When this job is activated (i.e., when the temporal clause of the condition is true), it will check the spatial clause of the condition resulting in one of the following three possibilities:
        i. The spatial clause of the condition is met, then the action is performed;
        ii. The spatial clause of the condition is not met and there is no further temporal margin for rule validity, then the rule is deleted;
        iii. The spatial clause of the condition is not met, but the temporal clause of the condition can be met at a later time, then the initial trigger will be set again (return to Step 1).

As an additional possibility, the checker job solution discussed earlier (see Section 4.2) is powerful enough to support more complex conditional expressions, despite its performance drawbacks. When a checker job is activated, it possesses the capability to check conditions of both spatial and temporal natures and, therefore, presents another comprehensive alternative to be taken into account. Finally, external control in the middleware is also capable of taking on conditional expressions of any nature and at any level complexity. While it represents a robust solution, it may, nevertheless, present problems where efficiency is concerned.

What follows in Section 5 is an evaluation of the alternatives presented both in this as well as previous sub-sections in order to determine the suitability and efficiency of each in particular cases.

5. EVALUATION OF SPATIO-TEMPORAL RULES

The goal of this section is to ascertain which of the implementation alternatives proposed is most appropriate for different workloads. All the experiments were run with the Oracle Database 11g on a dedicated Sun Fire X4150 server with two Intel Xeon Quad-Core L5310 processors (1.6 GHz, 4 GB RAM) and a SAS 104 rpm hard drive (~4 ms latency). Each experiment was repeated ten times for noise reduction. Results are presented as average values and range from minimum to maximum.

The experiments vary in the number of concurrent users (from 1 to 50) and of concurrent services for each (from 1 to 30). The focus of the results centres on (1) time delay from event occurrence to action execution and (2) Oracle's metric based on *consistent gets* (logical buffer I/O requests). The following experiment exposition is divided into three sub-sections, each corresponding to one of the active rules design cases presented in the previous section. A discussion of the results concludes the section.

5.1 Evaluation of Active Rules for Spatial Conditions

The two active rule implementation alternatives compared here are by (1) checker trigger based on a rules definition table (as described in Section 4.1) and (2) external control encoded in the middleware (in Java 1 6.0). The results for both implementation methods are shown in Figure 3.

As can be observed in these graphs, time delay is slightly lower for the checker trigger-based implementation. Additionally, external control involves a higher cost in terms of logical read

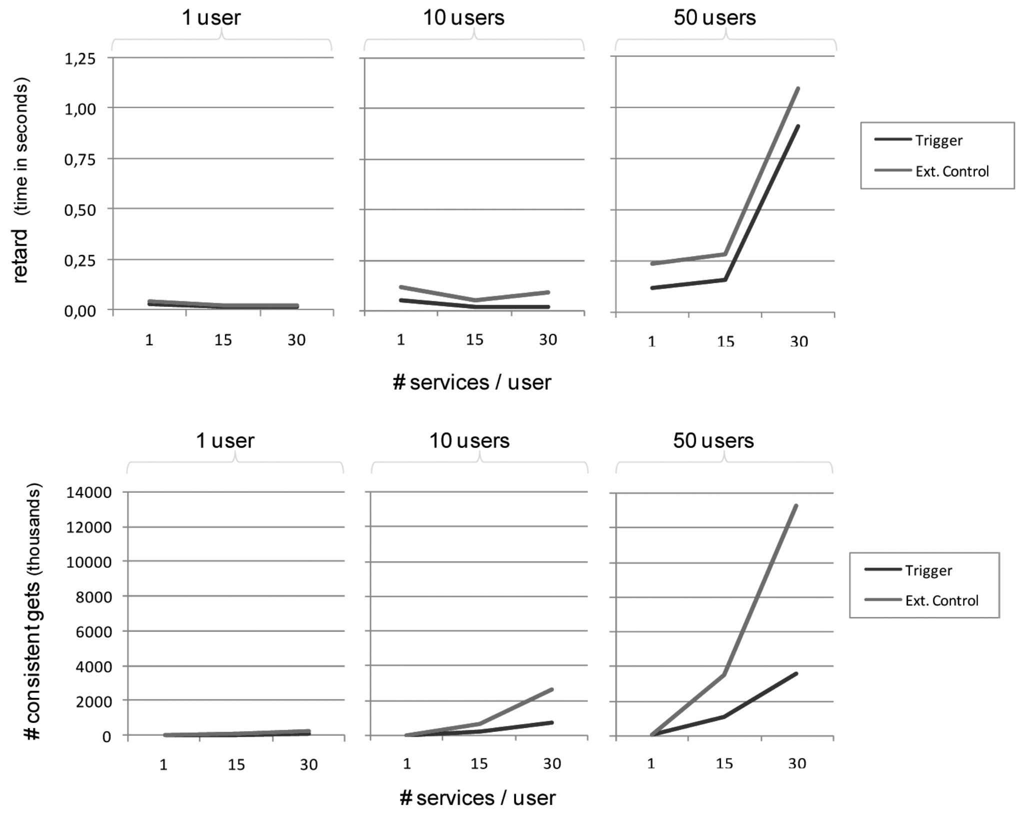

Figure 3: Comparison of implementations of active rules with spatial conditions

operations. It thus may be concluded that trigger mechanisms are more appropriate or the implementation of active rules with spatial conditions, especially when workload (i.e., number of concurrent users) is high. At the same time, however, it must be noted that external control is a highly scalable solution since workload may be distributed across several servers for the external control. As a result, the external control solution may also be considered an acceptable solution if there are enough hardware resources with respect to the number of concurrent users, since it demonstrates high performance for a low workload.

5.2 Evaluation of Active Rules with Temporal Conditions

In this evaluation, three implementation procedures are compared (see Section 4.2): (1) the assignment of a job to each individual rule, (2) the use of a rules definition table and a checker job and (3) by external control (including a timer with a granularity of one second). The second of these implementation alternatives is further divided into two sets of experiments, with a temporal granularity of one and five seconds, respectively. Results are presented below in Figure 4. Here, each solution loses efficiency at a similar rate as the workload increases. Nevertheless, the Job Manager alternative is more susceptible to the difficulty of balancing granularity and server collapse. The checker job with a granularity of five seconds produces less logical read operations,

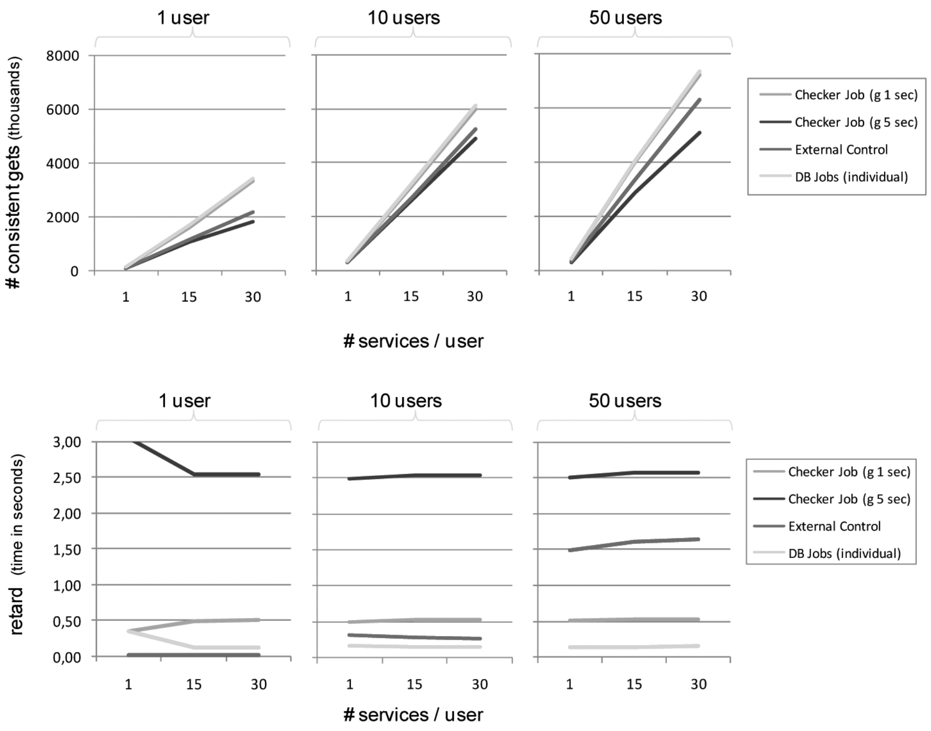

Figure 4: Comparison of implementations of active rules with temporal conditions

as demonstrated in Figure 4. If temporal granularity is too coarse, performance falls well below expectations (notice that for jobs with a granularity of five seconds, average delay is greater than 2.5 seconds, revealing low efficiency). However, if its temporal granularity is too fine and the workload is high, the server could jam and the efficacy of the service put at risk. Nonetheless, the individual jobs alternative involves nearly the same number of consistent gets as the checker job solution (where the latter had a granularity of one second) while producing shorter time delays, even with high workloads. In fact, the individual jobs implementation solution was only slightly affected in terms of time delays by high workloads.

Finally, external control shows balanced behaviour with even lower time delay than the individual job solution for low workloads. Since, as explained in the previous section, the external control solution is scalable, it may be considered a viable solution where there are enough hardware resources with respect to workload requirements. In such a case, each middleware resource would be assigned very few users.

5.3 Evaluation of Active Rules with Spatio-temporal (Mixed) Conditions

In this paper, three possible implementation alternatives were proposed: (1) using interrelated triggers and the Jobs Manager (see Section 4.3), (2) using the checker job (see Sections 4.2 and 4.3) and (3) with external control. Again, the checker job solution is evaluated twice at the granularity of one and five seconds, respectively. Results are compared below in Figure 5. The external control increases the logical I/O dramatically while maintaining a reasonably efficient performance with low delay times, except in cases of high workloads. In these latter cases, when the number of users per server is high, average delay time could fall outside the acceptable range. Conversely, while Jobs Manager is best with regard to the *consistent gets* metric, it offers lower efficiency than the other two alternatives proposed and particularly low efficiency with high workloads. As depicted in Figure 6, the checker jobs alternatives are not effective. Given that the validity of an active rule is met at a time slice, the execution of the corresponding action must be immediate in order to avoid losing validity due to the delay.

That is to say that if the condition is too restrictive, while the action is always executed, its execution is performed too late. In another group of experiments carried out, the spatial clause of a condition was a point query on a moving object (randomly relocated every five seconds) and the temporal clause was a wide range (five minutes). The spatial condition was met, but only once in the temporal range. A service was labeled a failure if the corresponding action was executed after the condition had become false (i.e., if the moving object was no longer at the point queried).

As depicted in Figure 5, loss of effectiveness is apparent for checker jobs with 50 users and 15 services requests. While such a loss of efficacy also affected other implementation alternatives at higher workloads, it was scarcely an issue for the external control solution (just four services among 1,500 requests) and it simply did not occur with the implementation of the interrelated trigger and Job Manager. It may be concluded, therefore, that the checker job is not an appropriate method for systems in which very restrictive conditions are to be applied. Finally, and with respect to the interrelated trigger and Job Manager proposal, not only does it offer effectiveness, but it also demonstrates high performance with low delay time (despite a somewhat high number of logical I/O requests) and good stability (maximum deviation from an average found to be below 0.5%).

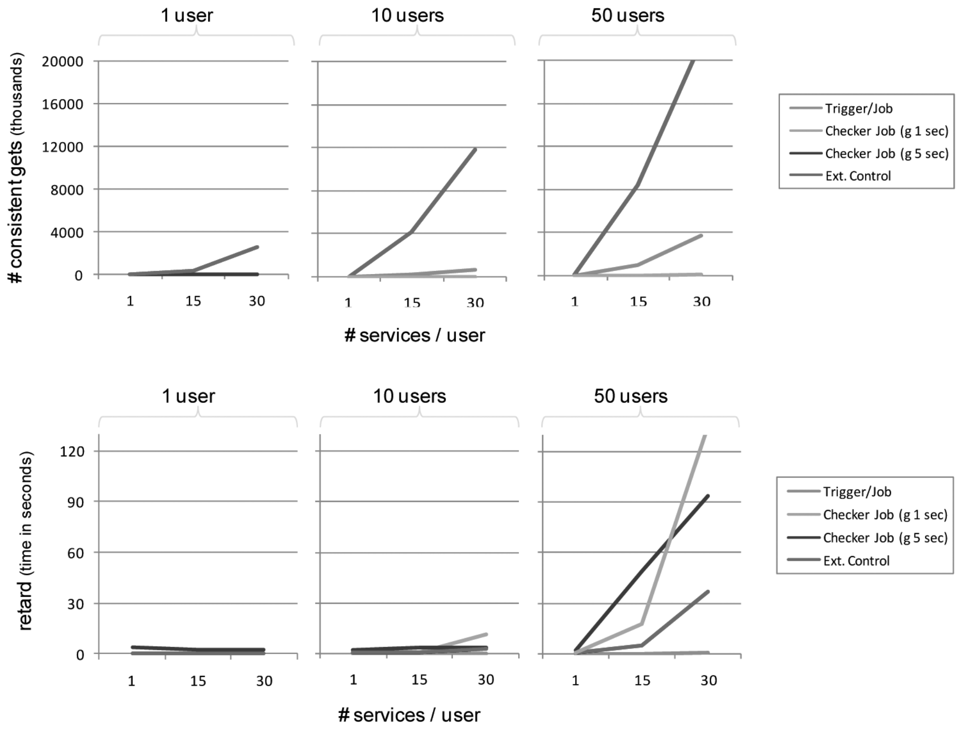

Figure 5: Comparison of implementations of active rules with spatio-temporal (mixed) conditions

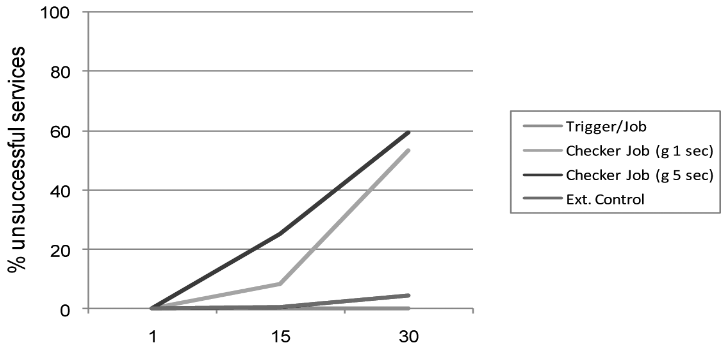

Figure 6: Service failure for point queries through different workloads (50 concurrent users, 1/15/30 services each)

5.4 Results Discussion

For the implementation of active rules with spatial conditions, it was found here that trigger mechanisms represent the most adequate solution. Events in the proper tables may be quickly detected, offering effectiveness and efficiency, provided that spatial information changes require database manipulation. This is not the case when spatial information is stored as trajectories or trends, for which spatial changes are assumed in time without altering the state of the database. For these cases, it is necessary to perform periodic checks by means of external control or the Jobs Manager.

If a temporal condition is combined with a spatial condition, the most efficient solution was found to be the setting of a trigger for the latter condition which, when executed, would activate a later job, and vice versa, until both conditions were met. Limits on the effectiveness of this alternative were not discovered in the experiments performed. For solely temporal conditions, the Jobs Manager implementation alternative proved to be effective at a minimum charge. The creation of a job for each rule was found to be the most efficient solution, yet such an approach has functional restrictions (regarding the complexity of the condition). This mechanism may also be used to create a solution based on rules storage and a checker job with comprehensive, periodic inspections. Such a solution would be functionally powerful, but lacks efficiency and even robustness – when workload is high, the processing of some events may be quite late. Such drawbacks may be minimized by increasing the resources used (i.e., splitting the workload amongst several checker jobs).

In addition, when rushed execution of active rules is not required (i.e., where the delaying of action execution by several seconds is permissible) the checker job with coarse granularity (five seconds) is a good solution, with a lower cost in I/O operations.

Finally, while the importance of external control proposal appears to be ranked lower by the experiment results, it is nevertheless a very interesting solution due to its scalability. In this paper, all experiments were performed over a single external control (i.e., middleware) server. Even so, the solution performed well with one user and even relatively so with ten users. While a high number of concurrent users proved difficult for the external control alternative, several servers could be set up to share the workload (e.g., by object assignment). Such an approach constitutes a robust and efficient alternative, yet it could require a good deal of hardware resources.

6. ILLUSTRATION OF THE SPATIO-TEMPORAL RULES APPLICATION

The proposal presented above has been applied in the implementation of a Situation Model for Natural Interaction (Cuadra *et al*, 2008) (hereinafter, the "Interactor System"). Such a system seeks to imitate human behaviour through interactive processes, taking into account diverse types of knowledge and reasoning mechanisms. Among these, the Situation Model deals with the circumstances of an interaction. The function of the model is to manage feasible circumstances, fix current ones, trigger warnings when a circumstance is attained (or when one is impossible to achieve) and to establish plans for achieving situation goals. One of most important aspects of circumstances is the managing of spatio-temporal information with the general aims described above. The spatio-temporal databases' support is most adequate because the physical devices employed to detect the position of users (e.g., wireless, RFID, etc.) can be diverse and are usually distributed; the environment can be subject to physical transformations (e.g., the closing of a parking lot when it is at its full capacity); and the interaction itself is a process that can be distributed across several users. Additionally, there exists a good deal of information on warning triggers which have been widely studied.

The Interactor system has a Situation Model supported by a spatio-temporal database over the Oracle RDBMS 11g. In a research project, the Interactor was applied in the setting of a hotel in which the physical layer of location was based on RFID technology. In this particular case, active rules for spatial conditions were implemented by means of checker triggers (e.g., for warnings such as "You are entering the wrong restroom. Proceed to the men's restroom."), active rules for temporal conditions were implemented through the Jobs Manager (e.g., for services such as "Wake me up tomorrow at seven o'clock.") and active rules for mixed, spatio-temporal conditions were implemented with an interrelated trigger and job (e.g., for warnings such as those given when arriving to the hotel's restaurant and it is past eleven o'clock).

As mentioned earlier, another requirement of the Situation Model is to establish plans. Building plans regarding the material aspects can be seen as a guiding problem (from a starting node to a goal node in a graph). The Interactor System builds a plan and applies it through active rules: to redirect a user at an intersection, to warn a user if he/she leaves a marked route or area and to state the success of the plan. Of course, the user may choose to eliminate these rules – hence, cancelling the plan – or adapt them – thus, altering the plan – at any moment as required. Hybrid active rules are also added for mixed, spatio-temporal goals (e.g., "Take me to the drawing room before eleven o'clock"), triggering warnings at particular points if the success of the plan at risk (e.g., "You should hurry or we won't get there on time.").

The Interactor System, with the described Situation Model, has been utilized in several research projects. While the physical layer in these projects was supported by RFID (as shown below in Figure 7), it is general enough to be supported by different location-aware devices. The database server maintains the spatio-temporal database, the knowledge bases and a multi-agent platform for communicating the system's components.

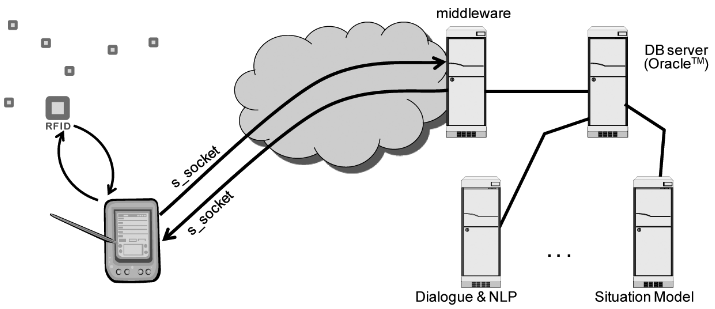

Figure 7: *Interactor* application architecture

7. CONCLUSIONS AND FURTHER CHALLENGES

In this paper, a practical approach to the implementation of active rules in spatio-temporal databases has been presented. The implementation proposal is supported by an evaluation, revealing that classic triggering mechanisms are the most appropriate and efficient in the majority of cases for active rules with solely spatial conditions. Similarly, the Oracle Jobs Manager is the appropriate alternative for rules regarding strictly temporal conditions. Finally, when rule conditions contain mixed, spatio-temporal clauses, a hybrid solution of interrelated triggers and jobs worked best.

The implementation based on a rule definition table facilitates the definition and inclusion of new rules. Description of ECA rules can be carried out in PL/SQL code. Furthermore, a simple tool (in Java) can show the definition of current rules, modify rules or add new rules, since all are basic database access operations. Since many domain experts are, oftentimes, not familiar with data languages, an interesting challenge for the future would be to perform a Natural Language-based interface for the tool, so that no specific knowledge would be required to manage existing rules or include new ones.

Furthermore, the rule definition table approach allows for the definition of dynamic rules; that is, rules which, as a result of the execution of their corresponding action, cause another rule to be included, activated, deactivated, modified or deleted (and even may cause the change of the original rule itself). However, these approaches have serious drawbacks in cases of precise temporal conditions and high workloads. For each different application, one must determine which mechanisms are best suited to the former's specific features and, where the requirements of functionality and performance appear simultaneously; find a compromise between the two.

8. ACKNOWLEDGEMENTS

This study was partially funded through two competitive grants awarded by the Spanish Ministry of Education and Science through the project THUBAN (TIN2008-02711) and the Spanish Ministry of Industry, Tourism and Trade through the project Semants (TSI-020100-2009-419).

REFERENCES

ABRAHAM, S. and SOJAN LAL, P. (2008): Trigger based security alarming scheme for moving objects on road networks. C.C. Yang *et al* (Eds.): ISI 2008 Workshops, LNCS 5075, 92–101.

AL-JUMAILY, H.T., CUADRA, D. and MARTÍNEZ, P. (2008): OCL2Trigger: Deriving active mechanisms for relational databases using model-driven architecture. *Journal of Systems and Software*, 81: 2299–2314.

AL-JUMAILY, H.T., CUADRA, D. and MARTÍNEZ, P. (2002): An execution model for preserving cardinality constraints in the relational model. *4th Int. Conf. on Enterprise Information Systems*, 819–822).

BERTINO E., CUADRA, D. and MARTÍNEZ, P. (2005): An object-relational approach to the representation of multi-granular spatio-temporal data. *17th Int. Con. in Advanced Information Systems Engineering Proceedings (CAiSE),* Porto, Portugal 119–134.

BERTOLOTTO, M. (1998): Geometric modeling of spatial entities at multiple levels of resolution. PhD Thesis, Università degli Studi di Genova.

CHAKRAVARTHY, S., KRISHNAPRASAD, V., ANWAR, E. and KIM, S. (1994): Composite events for active databases: Semantics, contexts and detection. In *Proc. of the 20th international Conference on Very Large Data Bases*.

CHEN, Y., RAO, F., YU, X. and LIU, D. (2003). CAMEL: A moving object database approach for intelligent location aware services. In *Proc. of the 4th Int. Conf. on Mobile Data Management, MDM 2003*, Melbourne, Australia, January 21–24.

CUADRA, D., RIVERO, J., VALLE, D. and CALLE, J. (2008): Enhancing natural interaction with circumstantial knowledge. *Trans on Sys Sci & Apps*, 4(2): 122–129.

DÍAZ, O., PIATTINI, M. and CALERO, C. (2001): Measuring triggering-interaction complexity on active databases. *Information Systems*, 26(1): 15–34.

EISENBERG, A., MELTON, J., KULKARNI, K., MICHELS, J. and ZEMKE, F. (2003): SQL: 2003 has been published, ACM SIGMOD Record, 33(1), March 2004.

GÜTING, R.H., BÖHLEN, M.H., ERWIG, M., JENSEN, C.S., LORENTZOS, N.A., SCHNEIDER, M. and VAZIRGIANNIS, M. (2000): A foundation for representing and querying moving objects. *ACM Trans. Database Sys*. 25(1): 1–42.

HEIMRICH, T. and SPECHT, G. (2009): Enhancing ECA rules for distributed active database systems. *Web, Web-Services, and Database Systems*, 2593: 199–205.

KAIYIN, H.K., PENGFEI, C.P., YI, C.Y., SONG, W. and CHEN X. (2009): The research for embedded active database based on ECA rule and implementation in SQLite database. *2009 First Int. Workshop on Database Technology and Applications*.

KEENEY, J., STEVENS, C. and O'SULLIVAN, D. (2010): Extending a knowledge-based network to support temporal event reasoning. In *Proceedings of the 12th IEEE/IFIP Network Operations & Management Symposium* (NOMS 2010), Osaka, Japan, 19-23 April.

KOUBARAKIS, M., PERNICI, B., SCHEK, H-J., SCHOLL, M., THEODOULIDIS, B., TRYFONA, N., SELLIS, T., FRANK, A.U., GRUMBACH, S., GÜTING, R.H., JENSEN, C.S., LORENTZOS, N., MANOLOPOULOS, Y. and NARDELLI, E. (2003): Spatio-temporal databases: The CHOROCHRONOS approach. Springer-Verlag, *Lecture Notes in Computer Science* 2520.

NORMAN W.P. and DIAZ, O. (1999): Active database systems. *ACM Computing Surveys*, 31(1): 1999.

ORACLE DATABASE 11G (2007): Enterprise scheduling. White paper, July 2007. http://www.oracle.com/technology/products/dataint/pdf/scheduler.pdf

ORACLE DATABASE 11G (2009): Complex event processing with rules manager. White paper, Sept. 2009. http://www.oracle.com/technology/products/database/rules_manager/pdf/twp_appdev_rulesmanager_11g.pdf

QIAO, Y., ZHONG, K., WANG, H. and LI, X. (2007): Developing event-condition-action rules in real-time active database. In *Proceedings of the 2007 ACM Symposium on Applied Computing* (Seoul, Korea, March 11–15, 2007). SAC '07. ACM, New York, NY, 511–516.

TRAJCEVSKI, G., SCHEUERMANN, P., BRÖNNIMANN, H. and VOISARD, A. (2005): Dynamic topological predicates and notifications in moving objects databases. In *Proc. of the 6th Int. Conf. on Mobile Data Management* (Ayia Napa, Cyprus, May 09–13, 2005). MDM '05. ACM, New York, NY, 77–85.

BIOGRAPHICAL NOTES

Dolores Cuadra

Dolores Cuadra received the MSc in mathematics from Universidad Complutense of Madrid in 1995. In 1997, she joined the Advanced Databases Group, at the Computer Science Department of Carlos III University of Madrid, where she currently works as lecturer. In 2003, she obtained the PhD degree in computer science from Carlos III University of Madrid. Her research interests include advanced database technologies, spatio-temporal databases and their applications to situation management. She has been working in the Computer Science Department at Purdue University of West Lafayette (Indiana) for nearly a year, where she has applied her research in spatio-temporal database.

Jessica Rivero

PhD student Jessica Rivero got her degree in telecommunication engineering from Carlos III University of Madrid in 2005. In 2006, she joined the Advanced Databases Group, at the Computer Science Department of this university, where she currently works as assistant teacher. In 2007, she obtained the Master in computer science and technology. Her research interests include advanced database technologies, spatio-temporal databases, metaheuristics algorithms and their applications to Situation management. During her stay in IRIDIA group (CoDE department of the Université Libre de Bruxelles (Belgium)) she validated her research in spatio-temporal databases and metaheuristics.

Francisco Javier Calle

Francisco Javier Calle got his degree in computer science from the Technical University of Madrid in 1997. In 2000, he joined the Advanced Databases Group in the Computer Science Department at the Carlos III University of Madrid, where he is currently working as lecturer. In 2005, he obtained the PhD degree in computer science from the Technical University of Madrid, *with a thesis in the natural interaction research. Apart from this, his research topics are focused on dialogue modeling, intentional processing and joint action models, and cognitive components of the interaction. He has also been working in several European and national research projects regarding human-computer interaction.*