MANAGING UNCERTAIN TRAJECTORIES OF MOVING OBJECTS WITH DOMINO

Goce Trajcevski, Ouri Wolfson*, Cao Hu, Hai Lin, Fengli Zhang
Naphtali Rishe†
Database and Mobile Computing Laboratory
Department of Computer Science
Univeristy of Illinois at Chicago
Email: gtrajcev,wolfson,hcao,hlin,fzhang@cs.uic.edu

Key words: Databases and Information Systems Integration, Moving Objects Databases

Abstract: This work describes the features of the DOMINO (Database fOr MovINg Objects) system, which brings several novelties to the problem of managing moving objects databases. Our robust model of a *trajectory* captures the inherent parameter of *uncertainty* of the moving objects location, which impacts both the semantics of spatio – temporal queries and the algorithms for their processing. In DOMINO, we present a set of novel operators which capture the spatial, temporal and uncertainty aspects of a moving object. The operators are implemented as UDFs (User Defined Functions) on top of existing ORDBMS and can be used for answering queries and generating notification triggers. DOMINO's implementation, in which ORDBMS are coupled with other systems, seamlessly integrates several technologies: 1. existing electronic maps are used to generate the trajectory plan on behalf of a mobile user; 2. real-time traffic sources are used to automatically update the moving object's trajectories; 3. powerful (web-browser) GUI enables users to monitor and pose queries about objects.

1 INTRODUCTION

Location-based services have generated a tremendous amount of commercial and scientific research in the recent years. The motivation is due to two main market trends: 1.) rapid drop of prices[1] of cell phones; Personal Digital Assistants (PDA); GPS devices and services, and: 2.) a bulk of wireless applications launched by many companies targeted towards re-cooping the existing technology investments and maintaining the customer's loyalty, alongside with attracting new customers (e.g. location-aware content delivery, described below).

There are two global categories of location-based commercial services which involve management of moving objects. The first one involves *mobile workforce/fleet management, automatic vehicle tracking and locating, transportation management and support*. Here, the management of resources (people or systems) is combined with routes' scheduling and tracking, and the main concerns are: accuracy; optimal resource utilization; emergency handling; etc... The second category includes *location-aware content delivery*. For example, along with driving directions, the wireless user is sent a message with instant coupons as (s)he approaches a store. Alternatively, on demand, the wireless user may be given an information about nearest hospital, gas station, etc... Management of moving objects in location-based systems is also of interest to the military (Chamberlain, 1995). In the context of a *digital battlefield* application, one would be interested in queries like: *"retrieve all the enemy tanks which will enter region R within the next 20 minutes"*. As for the scientific research, where the subject is termed *Moving Objects Databases* (MOD), there is a large body of work on modeling and querying the locations of moving objects (Sistla et al., 1997; Vazirgiannis and Wolfson, 2001; Wolfson et al., 1999b); indexing schemas for efficient retrieval/update (Agarwal et al., 2000; Kollios et al., 1999; Pfoser et al., 1999; Tayeb et al.,

*Research supported by ARL Grant DAAL01-96-2-0003, NSF Grants ITR-0086144, CCR-9816633, CCR-9803974, IRI-9712967, EIA-0000516, INT-9812325.

†Research partly supported by NASA (grants NAG5-9478, NAGW-4080, NAG5-5095, NAS5-97222, and NAG5-6830) and NSF (CDA-9711582, IRI-9409661, HRD-9707076, and ANI-9876409).

[1]According to the predications of International Data Corp., the number of wireless subscribers worldwide should be 1.1 billion.

1998); efficiency of location update incorporating the uncertainty (Pfoser and Jensen, 1999; Sistla et al., 1999; Trajcevski et al., 2002).

1.1 Motivation

It has already been observed that the majority of the applications which deal with moving objects management had been developed in an ad-hoc fashion and that Database Management Systems (DBMS's) have the potential of providing a good foundation for the services outlined above (Chamberlain, 1995; Sistla et al., 1997).

Currently, the providers of location-based services fall into one of the following categories:

• Vendors which offer technology aimed at the consumer market for the wireless service providers. An example is *Signalsoft* (www.signalsoft.com), whose software runs on proprietary equipment, mainly telecommunication switches.

• Vendors for mobile workforce management like *eDispatch* and *American Mobile Satelite Corp.* (www.MobilePosition.com). The focus is on electronic forms and work-orders, and communication between mobile devices and corporate databases.

• Geographic Information Systems (GIS) vendors like *ESRI Corp.* (www.esri.com) and *MapInfo* (www.mapinfo.com). The single-user systems that they typically offer are lacking the real-time capabilities for managing moving objects.

• Transportation Industry vendors such as *@Road* (www.atroad.com) and *Kinetic Computer Co.* (www.kin.com) which provide fleet management capabilities (e.g. tracking, dispatching, billing, etc...).

A crucial capability in all the software systems which deal with location-based services is the management of the location information of the moving objects. For the purpose of updating the location information, these systems rely on devices on board of moving objects (mounted or carried by people) utilizing some positioning technology like: – GPS signal, transmitted from the device to the server via some wireless network; – transmission towers which use triangulation to compute the location of an object in a given network; – fixed sensors (e.g. at toll booths) which identify a particular moving object; – cell-id which identifies a cell in which a moving object is located.

Location management, as described above, relies on periodically generated updates of the form *location-time* points (x, y, t). The sequence of points is transmitted (and stored in)to some information repository (e.g. a database).

This method is commonly known as *point-location management* and has some severe drawbacks: 1.) the position of an object is unknown in between updates; 2.) If the exact location is required at a time different than update, the moving object must be contacted. This, however, is much more costly when one wants to retrieve the locations of *all* the objects at a particular time instance, because all the objects will have to be polled; 3.) As an implication, the cost VS precision trade-off is bad, in a sense that if an accurate picture of the moving object's location is desired, a lot of bandwidth and processing power will be consumed; 4.) The software development becomes cumbersome. Even if the existing DBMS are used, some queries are difficult to express in SQL (e.g. *"retrieve all the delivery trucks which are inside a given region R always between 5:10PM and 5:30PM*); 5.) the location stored in a database cannot always be identical to the real location of the moving objects (due to factors like communication latency and/or the inherent imprecision of the GPS). This yields an uncertainty parameter which affects both the modeling and querying, as well as the processing the queries.

1.2 Our main contributions

To address the issues listed in the previous section, we have developed the DOMINO (Database fOr MovINg Objects) system. There are two global categories of novel results in the DOMINO project which contribute to the problem of moving objects management.

The first one consists of results which have more theoretical flavor. We have a robust model of a *trajectory* of a moving object which is different from the existing models in the literature. Our model captures the spatial, temporal and uncertainty aspects of a moving object in a uniform fashion and it overcomes many drawbacks of the point-location management of moving objects. Moreover, we introduced a set of powerful operators for querying moving objects with uncertainty and developed algorithms for their processing (Trajcevski et al., 2002; Wolfson et al., 1999a). We designed a model for generating notification triggers which "fire" when a condition of a particular query is satisfied (e.g. *"notify me when I am at location L_1"*). We also have an event log which keeps track of the updates to the trajectories[2].

The second category of contributions is the very implementation of the DOMINO system. It demonstrates an integration of several existing technologies, as well as different sources of information/ data sets. We show how the existing maps information can be utilized to generate the trajectory of a moving object which, subsequently, is stored in an Object-Relational database along with its uncertainty. We put together an existing GIS software (ArcView (ESRI, 1996)) with an ORDBMS (IDS2000 (Team, 1999)) to construct a trajectory server. Web browsers enable end-

[2] A demo version of the DOMINO project is available at http://131.193.39.200/mapcafe/mypage.html

users to access and query the database of moving objects' trajectories with a powerful GUI. We couple a PDA with a GPS on-board a moving object, so that the object itself "knows" if/when it has deviated beyond the tolerance threshold (uncertainty parameter) from its trajectory (as the database server is "aware" of it) and generates the location updates only when needed. This is one special feature of our model – it strikes a balance between the modeling power (awareness of imprecision) and computational efficiency (transmission costs/bandwidth consumption). We also use the available real-time traffic information (www.ai.uic.edu) to automatically update the trajectories of the moving objects which are affected by traffic fluctuations. Lastly, using the event manager, we provide a VCR-like capability which enables the users to *replay* the behavior of a selected (sub)set of trajectories.

The rest of the paper is organized as follows. In Section 2 we briefly introduce our model of the trajectory and give an application-centric view of the DOMINO system. Section 3 describes the logical architecture and the components of implementation of DOMINO. In Section 4 we illustrate the features that DOMINO offers to the end-users. Section 5 positions our work with respect to existing results, and in Section 6 we conclude and propose directions for future work.

2 ACTORS/GOALS AND THE TRAJECTORY MODEL IN DOMINO

In this section we present the users categories and the system boundaries of DOMINO. We also describe the model of the trajectory that is used in our system.

There are two main categories of DOMINO users: of *wireless* users and *WWW* users.

We assume that each wireless user has a PDA and a GPS available on board the moving object. Initially, the user sends a set of 2D (x, y) points which (s)he intends to visit along the route, together with the stay time at each point and the uncertainty threshold r, to the server. Given electronic map information such the average speed per block and the traffic conditions, the server will generate an optimal trajectory[3] which it will transmit back to the user. The trajectory is sent back to the wireless user as a set of 3D points (2D geography + time), giving a complete specification of the moving object's trip.

This is illustrated in Figure 1. As mentioned, at the beginning of the trip the wireless user enters on the

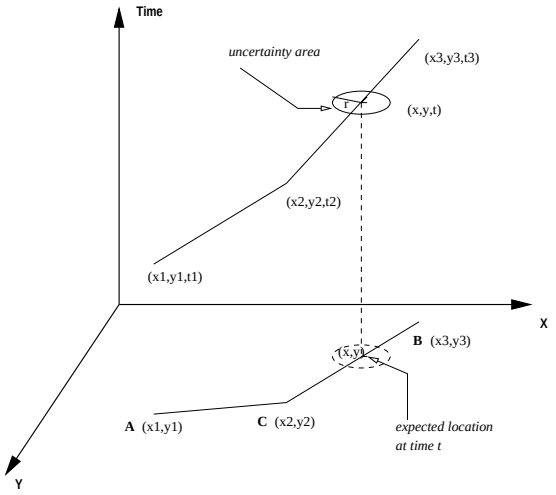

Figure 1: Trajectory of a moving object with location uncertainty, for a given route

PDA the sequence of 2D points like A and B on Figure 1. Along with the desired stay-times at each 2D point (the number of minutes that the user/moving-object will spend at each intermediate point), the user also provides an uncertainty threshold r. It indicates the maximum deviation allowed without a location update to revise the trajectory. Note that at any time point t, the uncertainty of the location of the moving object is a circle[4] centered at the object's *expected location* at t, with radius r. The server will calculate the optimal plan for the trip, which in Figure 1 is "via" the point C, and it will transmit back the sequence of 3D points (x_i, y_i, t_i), as shown in Figure 1 for $i = 1, 2, 3$. Each 3D point (x, y, t) describes either the geo-coded coordinates of a point along the route and the expected arrival time at that point, or the address along the route and the expected arrival time at that address.

During the trip, the PDA receives the locations' coordinates samples from the GPS. If at any time point t the location reported by the GPS is more than r units away from the expected location at t, the PDA will send a location update to the server. The server will re-calculate the *future* part of the trajectory and will transmit it back to the wireless user.

Observe that a mobile user always has a choice to update/modify its route at her/his own will. For example, it may introduce a new point "on the fly" which is not on the route, in which case the server will again generate a new (future) trajectory.

Figure 3 illustrates both categories of users and the

[3] Optimality may be based on a travel-time, shortest distance, or another criteria (e.g. load/unload time minimization)

[4] Recall that incorporating the uncertainty parameter in the model will affect the semantics (and the processing) of the queries posed to the DOMINO server. We defer this discussion to Section 4.

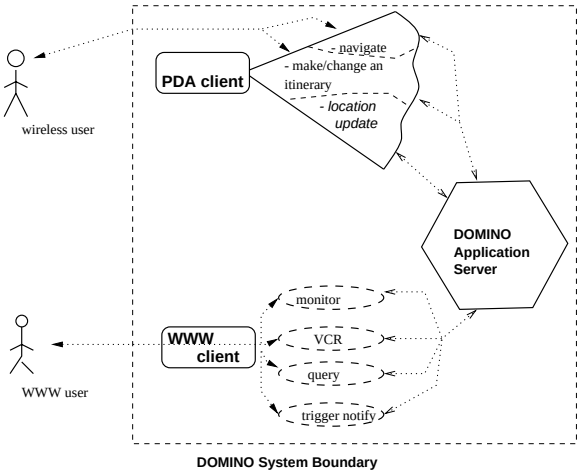

Figure 2: Actors and boundaries of the DOMINO system

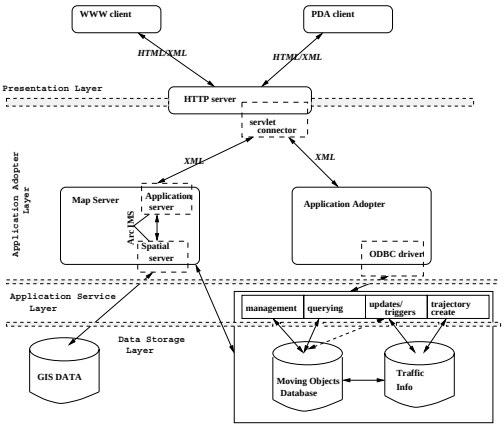

Figure 3: Architecture and components of DOMINO

system boundaries of DOMINO, blended with the use case contexts. Note how the PDA users do *not* intervene in order to generate location updates. These are generated automatically, whenever the PDA detects that the current location, as reported by the on-board GPS, deviates from the expected location by more than the uncertainty threshold r.

The second category of DOMINO users are the ones which use a web browser to access and query the system. Each WWW user has the ability to monitor a (sub)set of the moving objects from the database. (S)He can pose queries to the system and request that the system notifies her/him when a particular event occurs. The events occur at a time instance in which certain object(s) satisfy given condition, specified by the user. As we mentioned, there is also the VCR option which allows the WWW user to *Rewind* and browse the evolution of selected trajectories. We give a detailed description of these capabilities of our system in Section 4.

3 LOGICAL ARCHITECTURE OF DOMINO

In this section we describe the system structure of DOMINO at a general level, identifying the main components of the system and their relationships.

As shown in Figure 3, DOMINO has a multi-layered architecture which consists of subsystems organized in four layers.

• **Presentation Layer:** in which we have two subsystems, one for each category of users *wireless* and *WWW*, as we described in Section 2.

• **Application Adopter Layer:** Linked with the presentation layer via a *HTTP server*, this layer has two components:

1. The *Map Server* which provides the map utilities to the www user. It consists of two main subcomponents: – the *application server* which is the commercial ESRI's product ArcView (ESRI, 1996); – the *spatial server* which has been developed as a part of the DOMINO system.
2. *Application Adopter* which consists of a set of servlets that communicate with the main database server via ODBC drivers. Its main goal is to translate users' requests (e.g. query, trigger, etc ...) into a representation which can be understood and processed by the database kernel. Also, it translates the results into a form which can be presented back to the clients.

• **Application Service Layer**, as shown in Figure 3, is actually the set of different modules which provide the proper communication between the application adopter and the kernel level (data storage layer) for each particular service.

• **Data Storage Layer** is the kernel component of the DOMINO system. Here we have three main datasets: the GIS data; the real-time traffic data; and the Moving Objects (i.e. their trajectories) data. For each dataset, the kernel provides the functionalities needed by the higher-layer services. For example, the algorithms which process the operators that we described in Section 3 are implemented as User Defined Functions (UDF) on top of the IDS2000 ORDBMS. Also, the very definition of the *type* of the trajectory; their generation based on the PDA user's request; their update management (as well as events log); etc ..., are all part of the functionality of the Data Storage Layer.

4 FEATURES AND EXPERIMENTAL OBSERVATIONS

In this section we list the capabilities which DOMINO offers to its users. We also discuss part

of the experiments which we conducted in order to evaluate our model of the trajectory.

As we mentioned, introducing the concept of uncertainty will have an implication on the semantics of queries. In other words, it adds another "dimension" which needs to be considered when answering users' queries. Observe, for example, a particular moving object mo_1 and some static region R. Without uncertainty, the relative position of mo_1's trajectory with respect to R *and* with respect to a given time interval $[t_1, t_2]$ (i.e. spatiotemporal range query) would need to consider: 1.) Is mo_1 *inside* or *outside* R; and 2.) Does its relative position with respect to R hold *sometime* or *always* between t_1 and t_2. Once we allow the uncertainty of mo_1's location, one may also ask if: 3.) the relationships described by 1.) and 2.) above are *definitely* satisfied or they are *possibly* satisfied[5]. DOMINO offers several categories of features:

• **Point Operators** – this set of operators pertains to whereabouts of a moving object. The operators are:
When_At(oid, location), which returns a set of times at which a particular moving object oid is at the specified *location*. Note that the object may pass through the location more than once;
Where_At(oid, time), which returns the geographical location at which the object *oid* is expected to be at a given *time*;
When_Closest_to(oid, location), which returns the set of times at which the object *oid* is expected to be closest to a given *location*.

• **Range Operators** – these operators express the relative positions of a moving object's trajectory with respect to a static region, within a given time interval. The special "flavor" of this set of operators is due to the inherent uncertainty of the moving objects location, which is incorporated in our model. Each operator takes four input arguments: `(Trajectory, Region, Begin_time, End_time)` and returns *true* or *false*, depending if the given moving object's `Trajectory`, satisfies the relative position (expressed by the respective operator) with respect to the `Region` within the specified time interval. The set of operators consists of:
– *Possibly_Sometime_Inside* returns true if and only if the trajectory of the moving object is such that at some time *t* between `Begin_time` and `End_time`, the `Region` will intersect (or contain) the circle with radius r (uncertainty), centered at the expected location at *t*.
– *Possibly_Always_Inside* returns true if and only if there exists an "alternate route" for the moving object, such that: 1. for every time point $t \in [Begin\_time, End\_time]$, the distance between the expected location along the trajectory at t and the actual location on the alternate route at t does not exceed r; and 2. for every time point $t \in [Begin\_time, End\_time]$, the location on the alternate route at *t* is inside the `Region`.
– *Always_Possibly_Inside* returns true if and only if for every time point $t \in [Begin\_time, End\_time]$, the expected location along the moving objects trajectory is *no further* then r (uncertainty) from the `Region`. In other words, the circle centered at the expected location at *t* with radius r, will intersect the `Region` for every $t \in [Begin\_time, End\_time]$. Note the subtle difference between the last two operators. The former requires an existence of a *unique* alternate route, where as the latter does not. Semantically, the operators coincide for a convex `Region` (c.f. (Trajcevski et al., 2002)).
– *Definitely_Always_Inside* returns true if and only if the trajectory of the moving object is such that at every time point$t \in [Begin_time, End_time]$, the circle centered at the expected location at t with radius r is entirely contained within the `Region`.
– *Definitely_Sometime_Inside* To explain this operator, recall the meaning of the uncertainty. The moving object may take an alternate route (or experience some time fluctuations along its own route) which will cause him to deviate from its 3D trajectory. However, for as long as that deviation does not exceed the distance r from the expected location along the trajectory at time *t*, we "view" it as an "acceptable alternate trajectory" and we do not update the central server (i.e. its database). Thus, the intuitive meaning of the operator *Definitely_Sometime_Inside* is that it will return true if and only if no matter which *acceptable alternate trajectory* is taken by the moving object, it will intersect `Region` at some point between `Begin_time` and `End_time`.
– *Sometime_Definitely_Inside* returns true if and only if there exists a particular time point $t \in [Begin\_time, End\_time]$ such that the circle with radius r centered at the expected location at time *t* is entirely within the `Region`. In other words, no matter which *acceptable alternate trajectory* the object takes, at that particular t it will be inside the `Region`. Observe the subtle difference with the operator *Definitely_Sometime_Inside* in which we do not request that t is unique for all the possible *acceptable alternate trajectories*.

The set of operators described above has twofold usage. One is the "regular" *query* posed to the central server. The other usage which we have implemented in our DOMINO project is the

[5]Due to lack of space, we only list and intuitively explain the set of spatiotemporal operators which we introduced in our DOMINO project. Their rigorous semantics, as well as the algorithms which implement them is presented in (Trajcevski et al., 2002).

notification trigger. Namely, users have the option to request the DOMINO to *notify* them at the time when a particular operator becomes true.

Figure 4, which due to formatting needs is placed at the end of this article, depicts a snapshot of our GUI where a www user is posing a query. It presents the map of Chicagoland (Cook County) and three trajectories. Each of the trajectories indicates the stop – points and the respective stay times, as specified by the PDA user (the moving object). The shaded rectangle on the figure is the visual representation of the query region. As shown, DOMINO GUI offers a menu in which the user can select a query type. Observe also that the user enters the query time interval within this menu.

• **VCR** – a particularly interesting feature which DOMINO offers to the WWW users is the "playback". As the trajectories have been updated from their initial values, DOMINO keeps an event log. Using this log, it enables a visual representation of the "evolution" of the trajectories which the user can view and study their dynamics.

• **Cost-based optimization of the imprecision VS resource-consumption trade-off** – Controling the amount of uncertainty in a system incurs a cost. For example, if a moving object transmits its location to the database every x minutes or every x miles, then lowering x would decrease the uncertainty in the system, but increase bandwidth consumption and location-update processing cost, and vice versa. Adjusting the uncertainty threshold r in our trajectory model has the same tradeoffs concerning resource consumption. We developed a cost based approach to quantify this tradeoff. The information cost of a trip has the following three components: deviation cost, uncertainty cost, and communication cost. Using these costs we define a function that represents the overall information cost of a trip, and define the optimal uncertainty threshold as the value that minimizes this function.

• **Real-time traffic** – we utilized the fact that there exist sites which provide an information about the traffic conditions (e.g. congestions) along major expressways. These sites are maintained either by academic institutions (e.g. NTUA in Athens, updated every 5 minutes) or commercial companies (e.g. traffic map of Boston, maintained by SmartTraveler). In DOMINO, we use the traffic map of Chicago which is maintained by UIC (www.ai.cs.uic.edu) and is updated with current traffic information every 2 minutes. We utilize this information to identify trajectories which are affected by the traffic fluctuations and we automatically update them. Subsequently, the new trajectory information is sent to the respective moving objects.

Let us point out that we have conducted an extensive set of experiments with real data. We have constructed 1141 trajectories based on electronic maps of 18 counties in Chicago metropolitan area. Their sizes varied between 1 and 289 miles. We observed that the storage requirements (i.e. number of points/ segments) is linear with the length of the route of a trajectory. On the average, we needed 7.2561 segments per mile of a trajectory.

We have also performed driving tests to check the validity and robustness of our model. For example, we observed that with an uncertainty radius of 0.5 mile, we had about 5 updates per mile. Given the frequent fluctuations of a real traffic (stops; traffic lights; velocity variations), this verifies that our model and update policy indeed bring substantial savings in the number of updates, thus saving a lot of bandwidth/communication-cost compared to periodic updates.

5 RELEVANT WORK

There are two categories of results which can be compared with our DOMINO system. On the theoretical research side, the database community has been very active in investigating the properties needed by a DBMS which handles moving objects. One can identify few main streams: 1. *modeling and querying* – which has been addressed in (Guting et al., 2000; Sistla et al., 1997; Vazirgiannis and Wolfson, 2001; Wolfson et al., 1999b); 2. *Formal methodologies* – like the rich algebra of types and operators as presented in (Forlizzi et al., 1999; Forlizzi et al., 2000). 3. *Indexing techniques* – a set of works which tackle the most appropriate index structure (Agarwal et al., 2000; Kollios et al., 1999; Pfoser et al., 1999; Tayeb et al., 1998). Each proposed indexing method addresses different aspects of interest to management of moving objects. For example (Agarwal et al., 2000) addresses different variations of *nearest neighbor* query; (Pfoser et al., 1999; Tayeb et al., 1998) are concerned with the complexity of insertion/ deletion and updates, as well as the "aging" of the index (given the temporal aspect of the data). The work of (Pfoser and Jensen, 1999) considers a model of the uncertainty of a moving object. However, the model pertains to *past* trajectories, and the (2D) uncertainty region between 2 consecutive points is an ellipse.

A unique aspect of our approach is that we incorporate the (inherent) uncertainty parameter within the model of our trajectory; we consider the implication that it may have on the processing of user's queries; and we implement the update policy which considers the uncertainty of the moving objects location. We have also implemented the set of operators for querying uncertain databases, as UDF's on top of an existing ORDBMS.

The second category of relevant works, with respect to the DOMINO system, consists of the com-

mercial products which deal with location management (we have outlined some main representatives in Section 1). We note that none of the existing commercial products has utilized (and extended) the modeling and querying power of the ORDBMS. Since they lack the model of a trajectory, they do not utilize any information given in the electronic map to generate it, nor do they incorporate the real – time traffic information to update the moving object trajectory. Thus, our DOMINO system is unique in implementing novel research results, as well as integrating several existing technologies.

6 CONCLUSIONS AND FUTURE WORK

Throughout this paper we have introduced the main features of the DOMINO system and the novelties which it brings to the field of moving objects management. As we discussed, there are two categories of contributions. One has more theoretical flavor and introduces a model of a trajectory of a moving object which includes the uncertainty of the object's location. We developed a set of operators for querying the databases with uncertainty, as well as an update policy which automatically modifies the trajectories of the moving objects whenever they deviate beyond the uncertainty threshold. This, as we have also verified experimentally, saves a lot of computational cost and bandwidth consumption.

The other set of contributions brought by the DOMINO project is its implementation which integrates several technologies in a seamless manner. The available information in electronic maps is utilized to construct the trajectories of moving objects. The real – time traffic information is utilized to automatically update the trajectories affected by unexpected traffic conditions. PDA and GPS systems are blended together so that the users on board of moving objects can have their trip planned and updated. A powerful GUI enables a user of a web browser to monitor and query the database of moving objects. The existing ORDBMS have been "powered" with UDF's which implement our operators for querying the uncertain trajectories.

There are few immediate extension of the DOMINO project. We are working on integrating different (heterogeneous) sources of data for storing moving objects (e.g. Oracle, DB2 (Ora, 2000; Davis, 1998)), as well as different types of PDAs (e.g. Visor, HP) for the wireless user. A particularly challenging research topic that we are tackling is the issue of query optimization. Namely, each category of queries is better handled (faster) by a particular kind of an index. However, in the presence of UDFs, the *refinement* time in the main memory (once the data is brought from the disk) is not negligible either.

REFERENCES

(2000). *Oracle8: Spatial Cartridge User's Guide and Reference, Release 8.0.4*. Oracle Corporation. http://technet.oracle.com/docs/products/oracle8/doc-index.htm.

Agarwal, A. K., Arge, L., and Erickson, J. (2000). Indexing moving points. In *19th ACM PODS Conference*.

Chamberlain, S. (1995). Model – based battle command: A paradigm whose time has come. In *Symposium on C2 Research and Technology, NDU*.

Davis, J. R. (1998). *Managing geo - spatial information within the DBMS*. IBM DB2 Spatial Extender.

ESRI (1996). *ArcView GIS:The Geographic Information System for Everyone*. Environmental Systems Research Institute Inc.

Forlizzi, L., Guting, R. H., Nardelli, E., and Schneider, M. (1999). A data model and data structures for moving objects databases. Technical Report 260-10/1999, Informatik berichte.

Forlizzi, L., Gutting, R. H., Nardelli, E., and Schneider, M. (2000). A data model and data structures for moving objects databases. In *ACM SIGMOD*.

Guting, R. H., Bohlen, M. H., Erwig, M., Jensen, C., Lorentzos, N., Schneider, M., and Vazirgiannis, M. (2000). A foundation for representing and queirying moving objects. *ACM TODS*.

Kollios, D., Gunopulos, D., and Tsotras, V. J. (1999). On indexing mobile objects. In *18th ACM PODS Conference*.

Pfoser, D. and Jensen, C. (1999). Capturing the uncertainty of moving objects representation. In *SSDB*.

Pfoser, D., Theodoridis, Y., and Jensen, C. (1999). Indexing trajectories of moving point objects. Technical Report 99/07/03, Dept. of Computer Science, University of Aalborg.

Sistla, A., Wolfson, P., Chamberlain, S., and Dao, S. (1999). Querying the uncertain positions of moving objects. In Etzion, O., Jajodia, S., and Sripada, S., editors, *Temporal Databases: Research and Practice*.

Sistla, A. P., Wolfson, O., Chamberlain, S., and Dao, S. (1997). Modeling and querying moving objects. In *13th Int'l Conf. on Data Engineering (ICDE)*.

Tayeb, J., Ulusoy, O., and Wolfson, O. (1998). A quadtree – based dynamic attribute indexing method. *The Computer Journal*, 41(3).

Team, I. D. (1999). Informix datablade technology: Transforming data into smart data. Informix Press.

Trajcevski, G., Wolfson, O., Zhang, F., and Chamberlain, S. (2002). The geometry of uncertainty in moving objects databases. In *EDBT*. to appear.

Vazirgiannis, M. and Wolfson, O. (2001). A spatiotemporal model and language for movign objects on road networks. In *MOBIDE*.

Wolfson, O., Jiang, L., Sistla, A. P., Chamberlain, S., Rishe, N., and Deng, M. (1999a). Tracking mobile units in real time. In *7th Int'l. Conf. on Database Theory (ICDT)*.

Wolfson, O., Sistla, A. P., Chamberlain, S., and Yesha, Y. (1999b). Updating and querying databases that track mobile units. *Distributed and Parallel Databases*, 7.

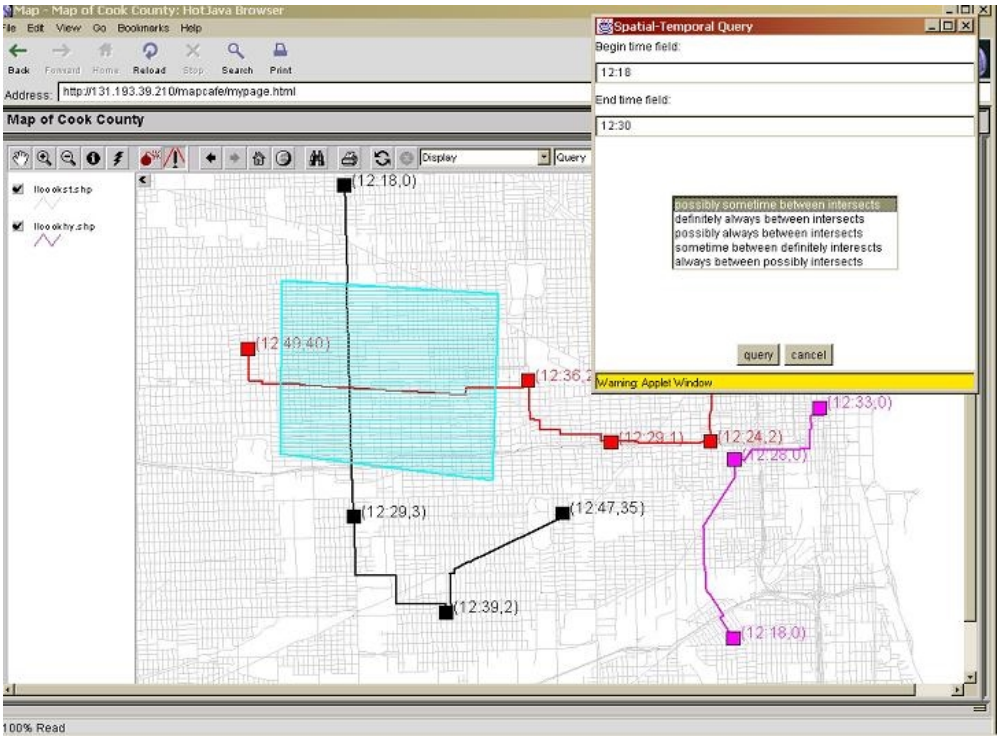

Figure 4: User posing a *Possibly_Sometime_Inside* query