FACTA UNIVERSITATIS (NIŠ)
Ser. Math. Inform. 18 (2003), 59–80

MODELING AND QUERYING MOBILE OBJECTS IN LOCATION–BASED SERVICES

Dragan Stojanović and Slobodanka Đorđević–Kajan

Abstract. A location–based service delivers geo–related information and geo–processing services to a mobile/stationary user in accordance with his current location and preferences or the location of mobile/stationary objects of his interest. Such services, like automatic vehicle location, fleet management, tourist services, transport management, traffic control and digital battlefield, are all based on mobile objects and management of their continuously changing locations. Thus modeling and the representation of mobile objects are fundamentally important in these applications. The goal of the work presented in this paper is to provide a mobile object data model for representing and querying mobile objects, with particular emphasis on those with point geometry, moving on a transport infrastructure that are pervasive in the location–based services. The mobile object data model extends widely accepted OGC Simple Feature model, also adopted by ISO TC 211. The implementation of the mobile object data model represents the foundation of the *m*STOMM component framework, intended for development of the location–based services that deal with tracking and management of mobile objects. As such, it is suitable for integration in any OGC compliant GIS platform or a DBMS, enabling them in the mobile object domain.

1. Introduction

The advances in the wireless communication technologies and mobile, Internet–enabled devices like the smart phones and PDAs have enabled global Internet connectivity and ubiquitous Web–based computing and service distribution. The recent convergence of Internet, wireless communications, mobile positioning and geographic information systems (GIS) has given rise to a new class of location–based applications and services. Location–based services (LBS) deliver geographic information and geo–processing

Received November 25, 2002.
2000 *Mathematics Subject Classification*. Primary 68P20.

power to mobile users in accordance with their current location and preferences, or to the stationary users in accordance to the location of the mobile/stationary objects of their interest. Location–based services are specialized, multi–tiered, component Web GIS applications, which can be published, located and invoked across the wireless/wired Web ([18]). Such services are useful in traffic control, road navigation, vehicle and person tracking, fleet management, tourist guiding and military exercises. During its motion, the user of LBS can express interest and issue queries related to the stationary objects (hotels, restaurants, roads, etc.), as well as the mobile objects (cars, passengers, tracks, etc.). The user may also represent the mobile object of interest to other mobile users within the same or another location–based service. Mobile objects that are of relevance in location–based services have point geometry and move along the predefined path in the transport network. To provide useful services to the mobile/stationary users based on their and locations of the surrounded objects, the LBS applications require database and application support to model and manage mobile objects in both database and the application domain, and to specify and process spatio–temporal queries with space and/or time conditions in respect to the actual time, to the past and to the (near) future. The main goal of our research work is to provide such support.

The paper is structured as follows. Section 2 presents the necessary support for the mobile objects in location–based services and specifies the fundamental model of the motion of the mobile point objects. Section 3 describes a standard–based, object–oriented mobile object data model, reviews its classes, attributes and operations, which represent a foundation for the mobile object query language and query processing. Section 4 sets the particular emphasis on the modeling mobile point objects that move either freely or over a predefined path in a network. Section 5 presents an implementation of the proposed mobile object data model in an object–relational database system with an existing OGC compliant spatial extension. Section 6 presents the overview of the related research work. Section 7 concludes the paper and identifies promising directions for the future research.

2. Mobile Point Objects in Location–Based Services

A location–based service is a specialized, multi–tiered Internet GIS application delivered across the wired/wireless Web to the stationary/mobile users ([18]) (figure 1). The location of the mobile devices in people's hands or built–in vehicles can be determined by using GPS, or mobile network triangulation. Their location is transmitted to the LBS server through the

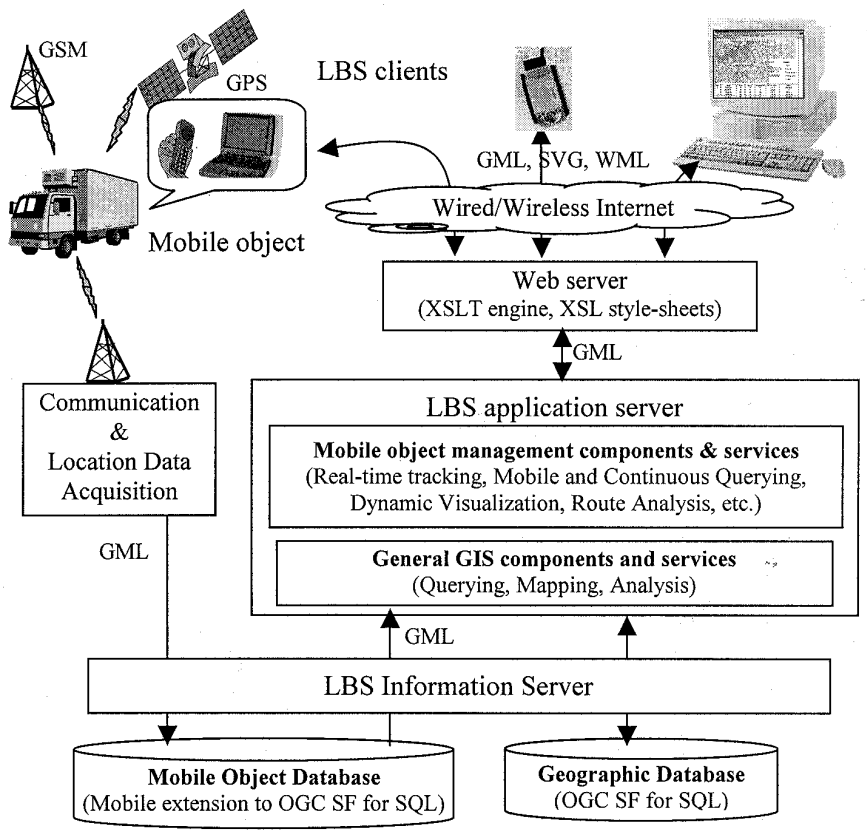

FIG. 1: Mobile objects in location-based services

wireless communication interface, from the device itself, or from a mobile positioning center of the mobile network. At the LBS server, the data is processed and services, based on such data, are provided to users. Mobile users may be also represented as the mobile objects registered at the server and tracked by others. Considering the location–based service applications we focus on the mobile objects with point geometry, whose size and shape are of no importance. Different components of the multi–tiered LBS architecture may use different mobile object data models for representation and management. Thus within the LBS application components, an object–oriented data model is implemented, while within the database system, relational or object–relational data model is implemented. To enable data communi-

cation between different application components, possibly implemented as Web services and distributed over the Internet, mapping between the different data models must be applied. With the advent of XML–based technologies in data transfer and exchange, it is reasonable to couple different data models by using a uniform representation of the mobile point objects. For the XML–based representation of the geographic features, OGC proposed an XML–based specification called GML (Geography Markup Language) ([10]). The GML definition must be extended for representing continuously changing properties of mobile objects.

According to T. Brinkhoff in ([1]) moving real–world point objects mostly move following a path in the network. Vehicles, trains, boats and passengers move following a particular network (roads, railways, rivers, pedestrian tracks). The air traffic also follows a 3–dimensional ($3D$) network of the air corridors. Even herds of animals often follow a (invisible) network path during their motion. In addition to this, mobile objects almost always know where they go, i.e. know their destinations. Also such mobile objects often use fast and/or shortest paths to their destinations depending on the cost criteria (time and/or distance). The application scenario for LBS that involves mobile objects both as the users of the service and as the tracked objects is as follows ([20]). The mobile object registers for a certain location–based service connecting to the LBS server by sending the starting location (the coordinates of the starting point or the address), the starting time, the ending location and eventually the set of points of interest that it is going to visit along its route and duration of the stay at each stop. The road network database accessed and managed by the LBS information server stores a network which is the layer supporting the object trajectories. Such road network database contains geometry for each road segment, the attributes used for geocoding, as well as the attributes needed for computing travel time and travel distance for a particular road segment, such as the average speed or the typical drive time ([21], [23]). The road network database can be updated by using real–time traffic information about accidents, roadwork, traffic congestions, etc ([22]). Given the starting motion parameters, LBS server calculates the shortest cost (distance or travel–time) path in the road network. During its motion, at regular or irregular time intervals, the mobile object sends to the LBS server a location update, containing the location value and the time instant when it is at that location. Using the sequence of location updates, the LBS server forms the mobile object's trajectory.

The trajectory of the mobile object is a polyline in three–dimensional space (two–dimensional space and time) defined by a sequence of points

(x_i, y_i, t_i) (figure 2). Such trajectory is only an approximation of the object's motion, because the object does not move in straight lines at constant speed. The number of points along the trajectory is proportional to the accuracy of such approximation. An additional parameter (mt_i) for every point defines the type of motion during a period $[t_i, t_{i+1}]$. The three motion types are defined: the *punctual* – the mobile object is not tracked and its location is not defined during a certain time period; the *stepwise* – the mobile object does not move during the time period; the *linear* – the mobile object moves along a straight line from (x_i, y_i) to (x_{i+1}, y_{i+1}), and at constant speed. When a mobile object moves along the curve segments with the variable speed, the fourth motion type, *interpolated*, must be defined to represent the motion by an interpolation function. These motion types correspond to the behavioral functions defined by Yeh and Cambray in ([25]).

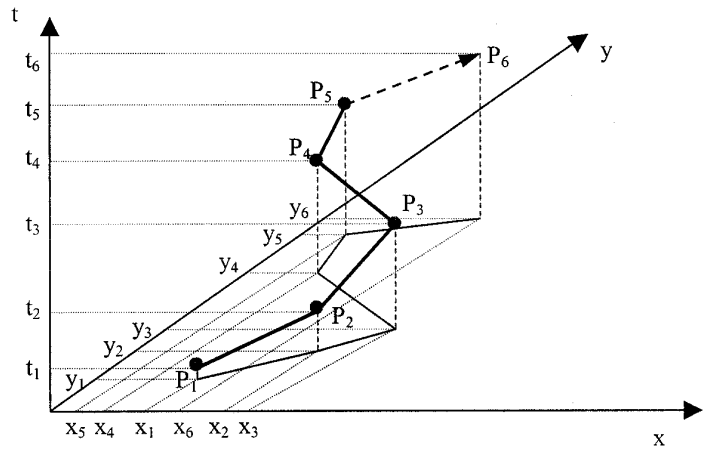

FIG. 2: Mobile point in a $3D$ space ($2D$ space, time)

Based on the trajectory of the mobile object, at any point in time t between t_i and t_{i+1}, the LBS server can compute the expected location of the mobile object at time t by using the linear interpolation. If the underlying road network database contains attributes needed for computing travel time on road segments, they enable the LBS server to define the expected trajectory of the mobile object starting from the last location update to the object's destination. If such attributes are not available, in order to define the expected trajectory of the mobile object, the LBS server needs the object's (average) speed at every location update. Upon registering to the service, the LBS server sends to the mobile object an uncertainty threshold.

The uncertainty threshold specifies the responsibility of the mobile object to send the location update to the LBS server if its current location is deviated from its expected location on the expected trajectory by the defined uncertainty threshold. With every location update, the expected trajectory of the mobile object from that location to the destination must be updated by $dt = ut/v$, where ut is the uncertainty throeshold and v is the average speed for the current road segment. When the route of the mobile object is not known a priori, because of, for example, the unknown destination, at any location update, the mobile object must send to the LBS server an additional attribute, its direction. It enables the LBS server to associate the mobile object to the road segment of the underlying network, and define and store a new part of the object's trajectory from the last to the new registered location, according to the shortest cost path between these two locations.

Considering the modeling and the representation of the mobile point objects several research problems arise. The data model and its implementation representations must include the whole trajectory of the mobile point object in the past, present and the (near) future and enable the estimation of its location at any time during the motion. The implementation of the data model must be compliant to the contemporary standards of OGC and ISO TC 211, that are widely accepted by both the industry and research organizations and incorporated in a wide variety of commercial products, such as Oracle 9i Spatial ([12]).

3. Modeling Mobile Objects in *m*STOMM Framework

The standard–based component framework, *m*STOMM (*mobile* Spatio–Temporal Object Modeling and Management), represents a suite of the mobile object modeling and the management techniques, tools and components, developed with the aim to support the development of the location–based services that involve mobile objects ([18]). The *m*STOMM data model was developed to support the conceptual modeling and querying of the discretely and continuously changeable properties of geographic features, i.e. mobile objects ([19]). The *m*STOMM data model is extensible, object–oriented, and specified using the UML class diagram notation. We base our modeling approach on the comprehensive framework of the data types and the rich algebra of operators defined in ([4], [5], [7]). We extend their approach to the object–oriented modeling paradigm, with respect to the OGC and ISO TC 211 standards, and provide the representation of the complete motion of the mobile object from the past to the future. We define the set of the mobile object data types and operations that are sufficient for the model-

ing mobile object's properties and expressing the queries based on them, in the real LBS implementations. Our main emphasis is put on the modeling the mobile point objects moving on the transportation network ([19]). Our goal is to develop the component framework, as a powerful and conceptually clean foundation for modeling and implementation of LBS applications that involve mobile objects on top of it.

The *m*STOMM data model appropriately extends the OGC Simple Features model, also accepted by ISO TC 211 ([8], [11]). The simple features are the geographic entities with the geometric properties represented by the collections of points represented by a pair of coordinates in $2D$ space, with a linear interpolation between the vertices. The standard defines abstract *Geometry* class, and its hierarchy of the specialized geometric classes (*Point*, *LineString*, *Polygon*, *MultiPoint*, etc.). The attributes and the operations, defined within the geometry classes, support the specification of the topological relations, direction relations, metric operations and the operations that support spatial analysis (point–set operations) on appropriate geometry types. The time dimension of a mobile object is specified through the *TimeObject* class hierarchy, defined in accordance with ISO TC 211 Temporal Schema ([9]) (*TimeInstant*, *TimePeriod*, *TimeDuration*, *MultiTimeInstant* etc.). The *TimeObject* class includes the attributes and operations for specifying topological relations, metric operations and point set operations on time dimension.

The base class for introducing mobility of the feature properties in the *m*STOMM data model is an abstract *MobileObject* class, as the root of the extensible hierarchy of classes for specifying mobile properties ([19]). Mobile attributes are represented as instances of classes, which inherit the *MobileObject* class and those are: *MobileGeometry*, *MobileNumber*, *MobileBoolean* and *MobileString* (figure 3). User–defined mobile classes of objects can be also specified by inheritance from the *MobileObject* class, or any of its specialized classes.

The *MobileObject* class defines general operations for the management and querying motion properties of the mobile objects, which are inherited and overridden in the specialized mobile classes. The *ObjectAt* operation retrieves an object's value at specific time instant. In order to restrict the sequence of motion slices of the mobile object according to the specific time object, the *ObjectDuring* operation is defined. That operation restricts the mobile object to only those motion slices from the sequence that belong to the specified time object given as an argument. The Lifespan operation returns a time object defining the whole life span of the mobile object. The

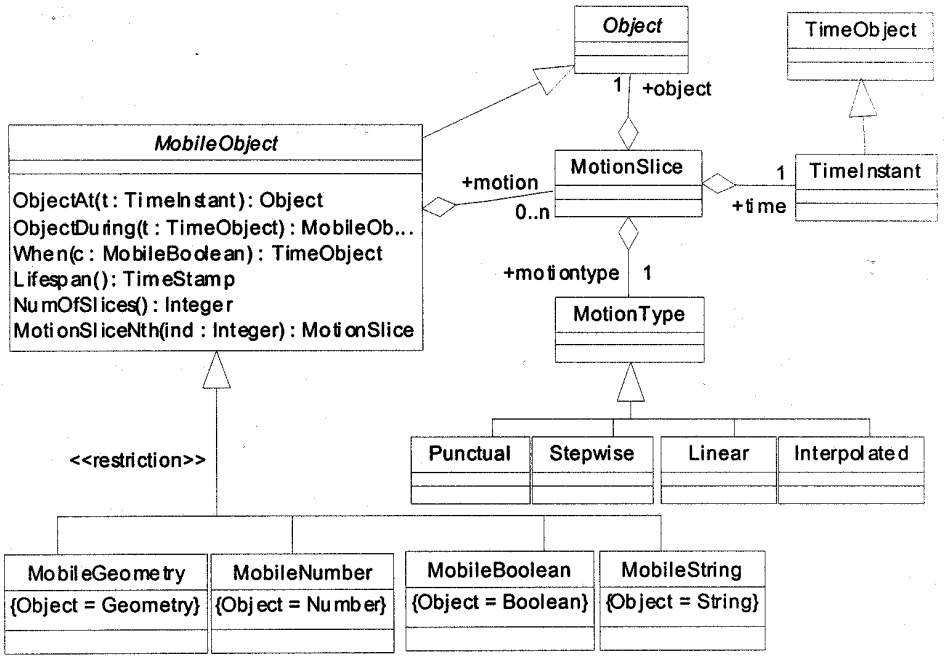

FIG. 3: Mobile object class hierarchy and associated classes

When operation returns the time object during which the mobile object satisfies the criteria specified by the mobile Boolean argument. The *NumOfSlices* and the *MotionSliceNth* operation enable navigation through the sequence of object's motion slices. The base class for modeling continuous change of geometric properties is an abstract *MobileGeometry* class, as the root of the extensible hierarchy of the classes for specifying mobile geometries (figure 4). The *MobileGeometry* class inherits both the *MobileObject* class and the OGC Geometry class. For every class in the Geometry class hierarchy an appropriate class for the representation of a particular mobile geometry is defined (*MobilePoint*, *MobileLineString*, *MobilePolygon*, *MobileMultiPoint*, etc.). Any of these classes appropriately restrict the Geometry class aggregated within the *MotionSlice* class (≪restriction≫ stereotype).

Being a specialization of the *Geometry class*, the *MobileGeometry* and its specialized classes can be treated in the same way as any other geometric object, i.e. it can represent the geometric property of any feature which is dynamic in nature and participate in all geometric operations and relations.

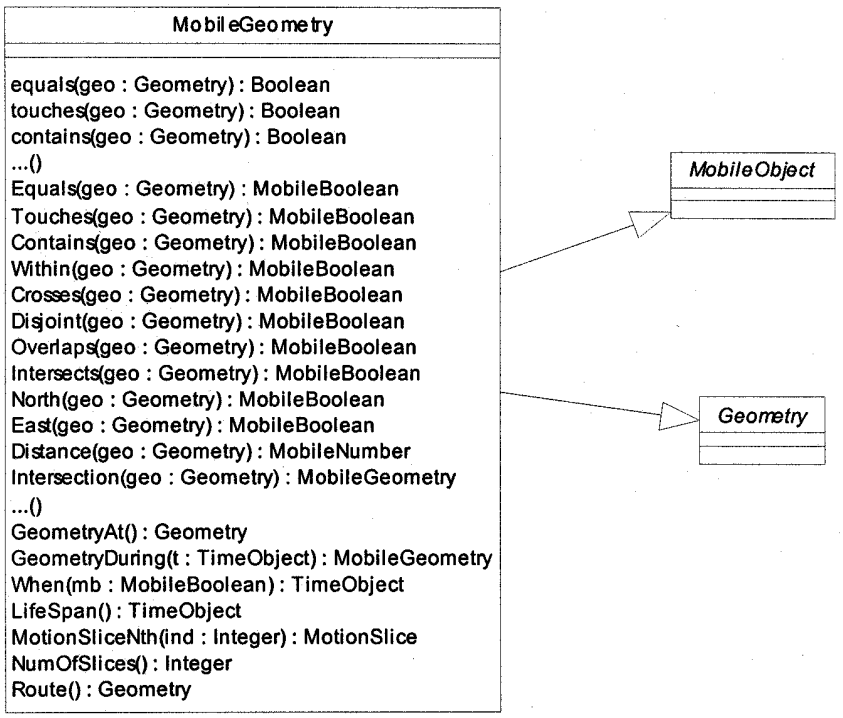

FIG. 4: The *MobileGeometry* class

An instance of the *MotionSlice* class, aggregated by the *MobileGeometry* class with multiplicity $0..n$, represents the registered location of the mobile geometry, by containing a geometry value (the instance of the *Geometry* class), the valid time of such value (the instance of the *TimeInstant* class) and the motion type (the instance of the *MotionType* class), which describes the way geometry changes between two successively registered geometries. The mSTOMM data model provides four classes for modeling motion types, and those are: the *Punctual*, the *Stepwise*, the *Linear* and the *Interpolated* class. Those motion types are comprehensive and can be used for expressing both discrete and continuous changes of spatio–temporal objects. The first three motion types do not require any additional information to be included in the data model, but for the interpolated motion type, specific interpolation parameters must be specified. The *MobileGeometry* class specializes the *ObjectAt* and the *ObjectDuring* operations defined in the *MobileObject* class, in the form of the *GeometryAt* and the *GeometryDuring* operations. Also the

new operation applicable to mobile geometry classes, the *Route* operation, is defined, which returns *Geometry* result representing the path traversed by the mobile geometry. The motion of the mobile object also causes continuous change of its non–spatial properties like the distance from some static or a mobile object or topological relation between two mobile objects. The *MobileNumber* and *MobileBoolean* classes, which inherit *Boolean* and *Number* base classes respectively, model those properties by aggregation *MotionSlice* class with appropriate restriction.

The *m*STOMM data model overrides topological relations, direction relations, metric operations and the spatial analysis operations inherited from the base Geometry class ([8], [11]). These relations and operations take a mobile or a non–mobile geometry argument and generate non–mobile results (*Boolean*, *Number*, *Geometry*, etc.). The specialized topological and direction relations, like the *touches* and the *contains*, have a single argument of type *Geometry* (which could also be the *MobileGeometry*) and return *Boolean* true value indicating that such relations are satisfied during the lifespan of the *MobileGeometry* argument(s) according to the temporal aggregation defined in ([5]). Such operations correspond to spatio–temporal predicates. Metric and spatial analysis overridden operations can not be considered as the predicates, and for the *MobileGeometry* argument the operation is delegated by default to the geometry value at the current time instant defined using *GeometryAt(Now)* operation. The *m*STOMM data model also defines mobile variants of the mentioned non–mobile relations and operations, named with the starting capital letter in figure 2. Such operations correspond to the temporally lifted operations that handle the non–mobile or the mobile geometry arguments and generate the mobile result, since for mobile argument(s) different results are generated in the course of time. The motion type associated with the mobile result depends on the operator or the relation. In general, it is not sufficient to inherit a motion type of a mobile argument. For example, for mobile points characterized by the linear motion type, a quadratic interpolation method is needed to represent the mobile result of the distance operation.

Our concept for finding the corresponding mobile operations and relations are similar to the operator lifting described in ([7]). For example, four variants for the touches operator existed, which is applicable to two geometric object and returns corresponding result:

$$\textit{touches: } \textit{Geometry} \times \textit{Geometry} \rightarrow \textit{Boolean}$$
$$\textit{touches: } \textit{Geometry} \times \textit{MobileGeometry} \rightarrow \textit{Boolean}$$
$$\textit{Touches: } \textit{MobileGeometry} \times \textit{Geometry} \rightarrow \textit{MobileBoolean}$$

Touches: MobileGeometry × *MobileGeometry* → *MobileBoolean*

As explained, the first and the second variants correspond to the *touches* operation defined in the *Geometry* class and overridden in the *MobileGeometry* class, and the last two variants is defined as the *Touches* operation in the *MobileGeometry* class and its specialized mobile classes. For clarity reasons, the names of the mobile operations and relations start with the capital letter, to be distinguished from their non–mobile counterparts.

To support the manipulation of mobile properties of type *MobileBoolean* and *MobileNumber*, predicates and operations (*and*, *not*, *max*, *add*, etc.) are overridden from the base classes (the *Boolean* and *Number* class respectively) and their temporally lifted counterparts (*And*, *Not*, *Max*, *Add*, etc.) are defined accordingly ([19]).

4. Mobile Point Objects on a Transport Network

As previously noted, the location–based services are concerned by the mobile point objects, i.e. objects with zero extent that change their location continuously over a predefined network infrastructure (roads, rivers, trains, air–corridors). Such objects are pervasive in many real–life applications. Also, mobile point objects constitute the simplest class of mobile objects so they can represent the foundation for modeling and representation of other mobile multi–point geometries with fixed or changing extent.

The *MobilePoint* class provides modeling mobile point objects that move continuously over a predefined network infrastructure (figure 5). Two attributes, the speed and the direction, are defined when the road network database does not contain travel attributes (average speed, travel time, etc.), or the route of the mobile object is not predefined (e.g. the destination is not known beforehand). The *MobilePoint* class inherits the *MobileGeometry* class and thus inherits and overrides the aforementioned spatio–temporal operations and relations.

The *GeometryAt* operation is overridden and implemented for the case that the underlying road network data does not contain the average speed or the travel time on specific road segments:

```
Geometry* MobilePoint::GeometryAt(TimeInstant t)
{
   int i = 0; double dist, path; float v;
   TimeDuration td;
   while (i < NumOfSlices () && MotionSliceNth(i)->
                                   timeinst.Before(t))
```

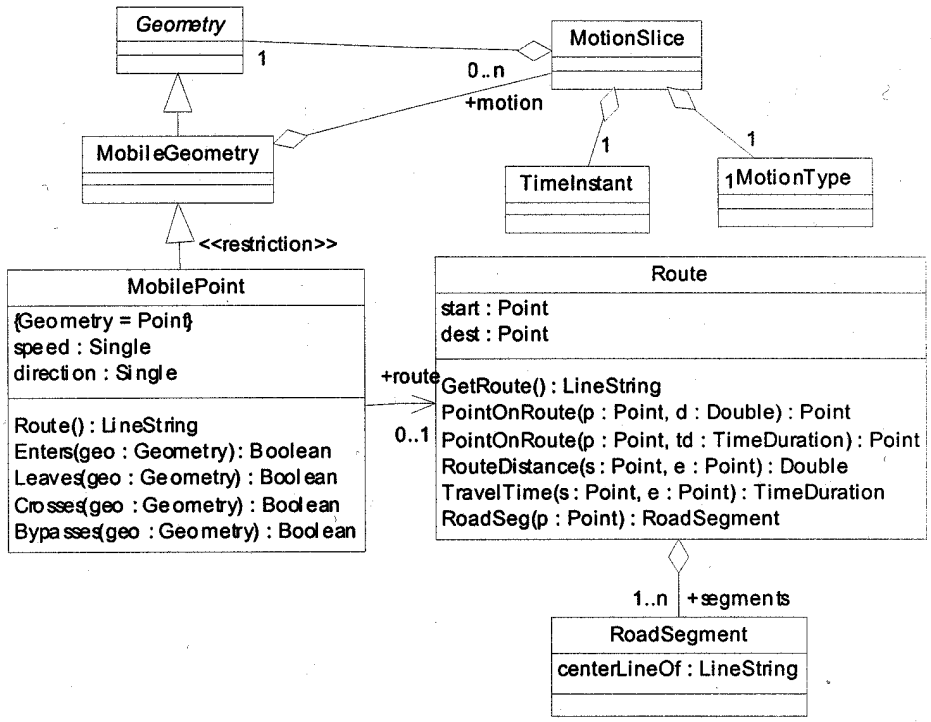

FIG. 5: Modeling the mobile point object in the mSTOMM data model

```
    i++;
int j = i; i--;
if(j < NumOfSlices())
{
    dist = route.RouteDistance(
            (Point *) MotionSliceNth(i)->Location(),
            (Point *) MotionSliceNth(i+1)->Location());
    td = MotionSliceNth(i+1)->Time() - MotionSliceNth(i)->
                                        Time();
    v = dist / td.Seconds();
}
else
    v = speed;
path = v · (t- MotionSliceNth(i)->timeinst). Seconds();
return route.PointOnRoute(
```

```
        (Point *)MotionSliceNth(i)->geometry, path);
}
```

The *Route* operation defined in the *MobileGeometry* class is redefined in the inherited classes, because for different mobile geometry types, the trajectory operation returns the specific geometric result. Such an operation applied to the mobile line string object yields the Polygon object as the result, and for the mobile point object with the associated linear motion function returns the *LineString* geometric object. The model defines operations arisen from the motion of a mobile point over a mobile or static polygonal area, such as *Enters*, *Leaves*, *Crosses*, *Bypasses*, etc. proposed in ([5]) as developments (figure 5). Such operations enable examination of changing spatial relations over time by simply sequencing basic spatio–temporal predicates. We extend the definition of those operations applying them to the mobile/static point and the linestring object using the same semantics and definitions given in ([5]) for the mobile point and the mobile region. Those operations are particularly useful in querying mobile points moving on network paths. Thus mobile object enters in the area of static or mobile polygonal object during given time period, if it was outside of the polygon at the beginning of the period (the *disjoint* relation), then at certain time instant was at the border of the polygon (the *touches* relation) and is within the region to the rest of the time period (the *within* relation). Thus, for a mobile point and a mobile linestring, the *Enters* predicate is expressed using a $C++$ syntax as:

```
Boolean MobilePoint::Enters(geo: Geometry)
{
   if(geo->GeometryType == "MobileLineString")
   {
      MobilePoint *int = (MobilePoint *) Intersection(geo);
      if(int->NumOfSlices == 0) return false;
      MotionSlice *touch = int->MotionSliceNth(1);
      TimeInstant *tt = touch->Time();
      TimeInstant *ts = LifeSpan()->Start();
      TimeInstant *te = LifeSpan()->End();
      return GeometryDuring(TimePeriod(ts, tt-1))->disjoint(geo) &&
            GeometryDuring(TimePeriod(tt + 1, te))->within(geo);
   }
}
```

The implementation of the *Enters* predicate uses the temporally lifted *Intersection* point–set operation applied to the mobile point and the mobile

linestring, yielding the mobile point as result. The spatio–temporal predicates disjoint and within are applied according to semantics defined in ([5]) as temporal aggregation. Similar definitions hold for *Leaves* (inverse of *Enters*), *Crosses* and *Bypasses* predicates. For the purpose of constructing an object's trajectory over a predefined route, the *Route* class is defined. The *MobilePoint* class associates the *Route* class which aggregates an ordered set of the *RoadSegment* class objects with an OGC SF *LineString* geometry, and contains the starting and the destination points that lie on the first/last road segment respectively (figure 5). The *Route* class defines operations for the determination of the point along the route on a specific distance or the travel time from the defined point (two versions of the *PointOnRoute* operation). For calculation the route distance or travel time between two points on the route (the *RouteDistance* and the *TravelTime* operations respectively). The *GetRoute* operation enables retrieval of the *LineString* object, which represents the route geometry, while the *RoadSeg* operation retrieves the road segment along the route, which contains the specified point.

5. Data Model Implementation in a Database System

The implementation of the proposed mobile object data model in an object–oriented application and a database is straightforward using the defined UML class diagrams. The data model implementation in the (object–) relational database domain is based on the definition of the user–defined data types and operations within object–relational DBMS and appropriate mobile extension of OGC Simple Features for SQL specification ([8],[11]). If the LBS server is based on a relational DBMS, the support for mobile object data management is integrated in the LBS application components. The user–defined data types and user–defined functions are defined in terms of the SQL DDL statement CREATE TYPE ([12]) as follows:

```
CREATE TYPE MotionSlice AS OBJECT
(slice# NUMBER, geometry Point, timeinst DATE,
mt MotionType);

CREATE TYPE Motion AS TABLE MotionSlice;

CREATE TYPE MobilePoint AS OBJECT
(GID NUMBER, speed NUMBER, direction NUMERIC,
motion Motion,
MEMBER FUNCTION GeometryAt(t TimeInstant) RETURN Point
...);
CREATE TYPE Ambulance AS OBJECT
```

(ID NUMBER, name CHAR(64), driver CHAR(64), type CHAR(256), locations MobilePoint);

CREATE TYPE Taxicab AS OBJECT
(ID NUMBER, company CHAR(64), driver CHAR(64), type CHAR(256), locations MobilePoint);

CREATE TYPE Street AS OBJECT
(ID NUMBER, name CHAR(64), centerLineOf LineString);

CREATE TYPE Municipality AS OBJECT
(ID NUMBER, name CHAR(64), area Polygon);

The operations defined for those types are implemented as the user–defined functions that can be applied to the user–defined data types. The query facility of SQL is provided by the well–known SELECT–FROM–WHERE clause. The user–defined operations on the user–defined data types can also be included as the predicates and the functions within the SELECT and WHERE clause and embedded in the SQL statement. Thus, the mobile spatio–temporal queries can be specified and processed, like:

1. Select ambulances that are within 2 *km* around of my current address:

select *amb.id, amb.name, amb.driver*
from *Ambulance amb*
where *(amb.locations.GeometryAt(Now)).distance(Point(xref,yref)) <= 2000*

We assume that the LBS application provides geocoding capabilities that convert location expressed as address in the point value with exact coordinates *xref* and *yref*. Thus the *GeometryAt* operation returns the location of the mobile point at the specified time instant (*Now* defines the current time). Operation *Distance* returns the distance between point values and the whole predicate in the WHERE clause specifies only those ambulances with such distance less then 2000 meters.

2. Return the length of path of truck "BioExport–1" in kilometers and time:

select *(t.troute.Route()).length(), (t.route.Lifespan()).duration()*
from *Truck t*
where *t.name="BioExport–1"*

The *Route* operation returns the *LineString* object and the *length* operation calculates the length of that line string. Similarly, the *Lifespan* operation returns the *TimeObject* object, and the duration operation applied on the *TimeObject* argument returns its duration.

3. Return the position of a taxicab "Banker–17" if it enters the "Cara Dusana" street in last 5 minutes:

select *t.locations.Geometry(NOW)*
from *Taxicab t, Street s*
where *t.name="Banker–17"* **and** *s.name="Cara Dusana"* **and**
(t.locations.GeometryDuring(TimePeriod(NOW–300,NOW)))
.Enters(s.centerLineOf))

The *Enters* operation applied to the *MobilePoint* object whose motion is restricted by the time period value which correspond to the "last 5 minutes" expression, and the *LineString* object representing the center line property of the street. The other two queries are self–explanatory.

4. Select all taxicabs that are currently in "Vozdova" street:

select *tcab.id, tcab.name, tcab.driver*
from *Taxicab tcab, Street s*
where *(tcab.locations.GeometryAt(NOW)).within(s.centerLineOf)*
and *s.name="Vozdova"*

5. Select all ambulances that were in "Niska Banja" municipality today between 5 and 6 AM.

select *amb.id, amb.name, amb.driver*
from *Ambulance amb, Municipality m*
where *(amb.locations.GeometryDuring(TimePeriod(5,6))).*
within(m.area) **and** *m.name="Niska Banja"*

On top of the *m*STOMM framework, we have developed a prototype application, named ARGONAUT for tracking, navigation and guiding mobile objects for tourist and business purposes. The ARGONAUT application enables testing the effectiveness of our proposal for mobile object's modeling and data management in a close–to–real conditions and environments using motion data generated by our GPS simulator ([20]).

6. Related Work

In the literature, the research work on modeling and querying of the mobile point objects can be categorized into the modeling and the representation of either the present and the (near) future information or the past information. The first research direction includes the works of Sistla et al. ([17]), Wolfson et al. ([24]) and Porkaew et al. ([14]) on modeling the current location of mobile points, including the prediction of the future trajectory with specified uncertainty, and update policies. Within the second research

direction, most approaches deal with the complete trajectories of the mobile objects ending with their last registered location ([4], [6], [7], [13]). Regarding the indexing and the query processing of the mobile point objects, Pfoser et al. ([13]) consider the whole trajectory of the mobile point object, while Saltenis et al. ([15]) and Porkaew et al. ([14]) consider only the present and the future motion of the mobile point object. Several research papers are related to the constrained motion of the mobile point object, over the path in the network and the predefined trajectory ([21], [16], [3], [1]). Vazirgianis and Wolfson in [21] deal with the mobile point objects whose motion is constrained by the road networks, with the known destinations. Their mobile point object data model completely relies on the comprehensive road maps that contains average travel time for road segments, and thus does not consider the real movement of the mobile object which reports its location according to the specified frequency or the uncertainty threshold. Shasabi et al ([16]) makes the same assumption by proposing efficient modeling and query processing of the mobile objects with predefined paths and schedules like trains moving on the rail network. Brinkoff ([1]) presents the generator of the mobile point objects following the given network intended for the generation of the suitable mobile point datasets required for benchmarking a spatiotemporal database. Chon et al. ([3]) model mobile objects that follow certain paths, characterized by the physical limitations (number of lanes, route conditions), and observe that the real trajectory of the mobile object is the result of the interactions among mobile objects in the system. They propose a partitioning approach rather than indexing to efficiently manage mobile objects' trajectories. They do not consider mobile data types and mobile query specification. Sistla et al. in [17] propose the Moving Object Spatio Temporal (MOST) data model that represents only the current and the expected location of the mobile point object, as continuous functions of time using so–called dynamic attribute, and the Future Temporal Logic (FTL) query language. The main issue addressed is how often motion vectors need to be updated to guarantee the error in calculated locations within specified threshold. This model does not offer a comprehensive set of mobile types and operations and does not consider motion that follows a specified network. Wolfson et al. in [22] propose a comprehensive approach to embedding moving objects support in a commercial DBMS according to MOST and FTL. Several issues related to implementation are addressed such as: the indexing moving objects, the update policies, the uncertainty regarding the location of an object, the dynamic behavior evolving with time, etc. The main issue is to determine how often updates of motion vectors are needed

to balance the cost of updates against imprecision in the knowledge of locations. Trajcevski et al. in [21] address the problem of querying moving object databases, which capture the inherent uncertainty, associated with the location of moving point objects. They present a set of spatio–temporal operators, which include uncertainty and are used to express spatio–temporal range queries.

A fundamental work for modeling and representation of trajectory of spatiotemporal objects, particularly the moving point and the moving region is the paper of Erwig et al. ([4]), where the moving object data types and the operations are presented and embedded into the query language. They do not propose a specific design of such types and operations, or a formal definition of their semantics. Gueting et al. in ([7]) continue this approach designing abstract data type extension to the DBMS data model and the query language. Besides the main types of interest, the moving point and the moving region, a relatively large number of auxiliary data types are defined, such as line type to represent a trajectory of the moving point and the moving real. The paper formally defines the abstract data types and operations, offering consideration on their design and embedding in the DBMS. Forlizzi et al. in ([6]) perform an adaptation and implementation of the abstract model presented in ([7]) by defining a discrete data model and proposing data structures for implementation within existing relational and object–oriented DBMS. They also do not consider movement along predefined routes on a network, or possibility to interpolate the motion between the location samples by high order polynomials or splines. Erwig and Schneider in investigates temporal changes of topological relationships between mobile objects based on an integrated view of space and time and on an algebraic data modelof spatio–temporal objects given in ([7]). They define a framework for the specification of spatio–temporal predicates that describe developments of well–known spatial topological relationships by building a sequence of elementary spatio–temporal and spatial predicates. In their paper Brinkhoff and Weitkaemper ([2]) propose a quite simple model of spatiotemporal objects where the location of mobile object is static unless is explicitly changed by the next sample location (the stepwise motion). They define an XML–based representation of spatiotemporal objects based on the proposed model, but without an adequate mobile extension to GML, as well as an architecture that supports such representation. They define continuous queries by which the clients require information about relevant updates in the spatio–temporal database.

None of the mentioned research work defines a fully object–oriented mo-

bile object data model including object–oriented mechanisms like inheritance, polymorphism, association, etc. Also, neither of proposed modeling solutions is compliant to the standardization of OGC and ISO TC 211 related to Simple Features ([8], [11]).

7. Conclusions and Future Work

The main contribution of this paper is the object–oriented data model and the SQL extension for mobile objects. The main features and benefits of our proposal are:

– Provides simple, but expressive mobile extension of OGC and ISO TC 211 Simple features specification for object–oriented modeling of the whole trajectory of the mobile point object, its past, current, as well as its future motion, which is either unconstrained or constrained by the network. It is part of the comprehensive data model for modeling general mobile objects, like the mobile numeric values, the mobile Boolean values and the mobile geometries with a non–zero extent and shape, like the mobile polygons. It specifies the mobile, spatio–temporal operations and relations as the foundation for the mobile object query language as extension to the SQL.

– Provides the foundation of the mSTOMM component framework implemented in MS Visual C++ development environment. Such framework enables any relational and object–relational DBMS or a GIS platform by mobile object data modeling and management capabilities. It is intended for easy and effective development of the LBS applications on top of it. The implementation of the mSTOMM data model in the Oracle 9i DBMS is currently in progress.

This work points to several directions of current and future research:

– The specification of the mobile GML extension as the implementation of the proposed data model, intended for the exchange of the data about mobile objects between the application components (Web services) residing on different tiers of a LBS application. For displaying mobile GML content, an appropriate XSL style sheet and transformation into the SVG format is going to be developed.

– The processing of the continuous location–dependent queries where not only the user issuing the query can change its location, but the objects involved in such query can move as well, and thus the query answer must be updated continuously.

– The improvement of the routing algorithm and the route computation

taking into consideration the actual, time–dependent traffic situation (traffic jams, road conditions, etc.).

Acknowledgements. This research was supported in part by Ministry of Science, Technology and Development, Republic of Serbia, and Municipality of Niš, under the project "Geographic Information System for Local Authorities based on Internet/WWW Technologies", Contract No. IT.1.23.0249A.

We wish to thank the anonymous referees for their valuable comments and suggestions pointing to us a way to significantly improve the paper.

REFERENCES

1. T. Brinkhoff: *A Framework for Generating Network–Based Moving Objects.* GeoInformatica Journal **2** (2002), 153–180.
2. T. Brinkhoff and J. Weitkaemper: *Continuous Queries within an Architecture for Querying XML–Represented Moving Objects.* In: Proceedings of SSTD 2001 Conference, 2001, pp. 136–154.
3. H. D. Chon, D. Agrawal and A. E. Abbadi: *Data Management for Moving Objects.* IEEE Data Engineering Bulletin **25**, No. 2(2002), 41–47.
4. M. Erwig, R. H. Gueting, M. Schneider and M. Vazirgiannis: *Spatio–Temporal Data Types: An Approach to Modeling and Querying Moving Objects in Databases.* GeoInformatica Journal **3** (1999), 269–296.
5. M. Erwig and M. Schneider: *Spatio-Temporal Predicates.* IEEE Trans. on Knowledge and Data Engineering **4** (2002), 881–901.
6. L. Forlizzi, R. H. Gueting, E. Nardelli and M. Schneider: *A Data Model and Data Structures for Moving Objects Databases.* In Proceedings of ACM SIGMOD Conference, Dallas, USA, 2000, pp. 319–330.
7. R. H. Gueting, M. H. Boehlen, M. Erwig, C. S. Jensen, N. A. Lorentzos, M. Schneider and M. Vazirgiannis: *A Foundation for Representing and Querying Moving Objects.* ACM–Transactions on Database Systems Journal **1** (2000), 1–42.
8. ISO/ TC 211 Geographic Information/Geomatics: *ISO 19125 – Geographic information – Simple feature access.* 2000.
9. ISO/ TC 211 Geographic Information/Geomatics: *ISO 19108 – Temporal Schema.* 2000.
10. Open GIS Consortium: *OpenGIS ® Geography Markup Language (GML) Implementation Specification, version 2.1.2.* OpenGIS Project Document Number 02–069, 2002.
11. Open GIS Consortium: *OpenGIS Simple Features Specification for SQL, Revision 1.1.* OpenGIS Project Document 99–049, 1999.

12. Oracle 9i Enterprise Edition: *Oracle Documentation Library.* Oracle Corporation, http://technet.oracle.com (2002).
13. D. Pfoser, Y. Theodoridis and C. S. Jensen: *Novel Approaches in Query Processing for Moving Object Trajectories.* In: Proceedings of Int. Conf. VLDB, 2000, pp. 395–406.
14. K. Porkaew, I. Lazaridis and S. Mehrotra: *Querying Mobile Objects in Spatio–Temporal Databases.* In: Proceedings of 7th SSTD 2001, Redondo Beach, USA 2001, pp. 59–78.
15. S. Saltenis, C. S. Jensen, S. T. Leutenegger and M. A. Lopez: *Indexing the Positions of Continuously Moving Objects.* In: Proceedings of ACM SIGMOD Conf. 2000, pp. 331–342.
16. C. Shahabi, M. R. Kolahdouzan, S. Thakkar, J. L. Ambite and C. A. Knoblock: *Efficiently Querying Moving Objects with Pre–defined Paths in a Distributed Environment.* In: Proceedings of the ninth ACM international symposium on Advances in geographic information systems (Atlanta, Georgia, USA, 2001), ACM Press, New York, 2001, pp. 34–40.
17. P. Sistla, O. Wolfson, S. Chamberlain and S. Dao: *Modeling and Querying Moving Objects.* In: Proceedings of the Thirteenth International Conference on Data Engineering, IEEE Computer Society, Washington, DC, 1997, pp. 422–432.
18. D. Stojanović and S. Djordjević–Kajan: *Location–based Web services for tracking and visual route analysis of mobile objects.* In: Proceedings of Yu INFO Conference, Kopaonik, 2002, CD ROM (Serbian).
19. D. Stojanović and S. Djordjević–Kajan: *Extending Database Technology to Support Location-Based Service Applications.* In: Proceedings of ICEST 2003 Conference, Sofia, 2003, pp. 399–403.
20. D. Stojanović and S. Djordjević–Kajan: *Generating Motion Data for Mobile Objects on a Transportation Network.* In: Proceedings of Yu INFO Conference, Kopaonik, 2003, CD ROM (Serbian).
21. G. Trajcevski, O. Wolfson, F. Zhang and S. Chamberlain: *The Geometry of Uncertainty in Moving Objects Databases.* In: Proceedings of the International Conference on Extending Database Technology – EDBT02 (Prague, Czech Republic, March 2002), Springer LNCS 2287, 2002, pp. 233-250.
22. G. Trajcevski, O. Wolfson, B. Xu, and P. Nelson: *Real–Time Traffic Updates in Moving Objects Databases.* In: 13th International Workshop on Database and Expert Systems Applications (DEXA'02) (September 2–6, 2002) Aix-en-Provence, France, pp. 698–704.
23. M. Vazirgiannis and O. Wolfson: *A Spatiotemporal Model and Language for Moving Objects on Road Networks.* In: Proceedings of 7th SSTD 2001, Redondo Beach, USA, 2001, pp. 20–35.

24. O. Wolfson, B. Xu, S. Chamberlain and L. Jiang: *Moving Objects Databases: Issues and Solutions.* In: Proceedings of 10th SSDB Conference, 1998, pp. 111–122.

25. T. S. Yeh and B. Cambray: *Modeling Highly Variable Spatio–Temporal Data.* In: Proceedings of 6th AustraliAsian Database Conference, 1995, pp. 221–230.

University of Niš
Faculty of Electronic Engineering
Department of Computer Science
P. O. Box 73
18000 Niš, Serbia
e–mail: dragans@elfak.ni.ac.yu
sdjordjevic@elfak.ni.ac.yu