Proceedings of the IEEE ITSC 2006
2006 IEEE Intelligent Transportation Systems Conference
Toronto, Canada, September 17-20, 2006

Towards Delivering Context-Aware Transportation User Services

René Meier, Anthony Harrington, and Vinny Cahill

Abstract—**ITS architectures encourage integration of individual intelligent transportation systems into comprehensive platforms and enable sharing of information across a wide variety of systems and services. This paper presents a spatial programming model that has been designed as part of the iTransIT ITS framework to provide a standardized way to build value-added transportation user services and ultimately to deliver contextual transportation information to users. The spatial programming model is based on a topographical approach to modeling space that enables services to use potentially overlapping spatial context to correlate independently defined distributed information. This programming model has been evaluated by building a context-aware service for multi-modal urban journey planning.**

I. Introduction

Integrating individual intelligent transportation system into comprehensive platforms is a key challenge faced by transportation authorities in the provision of transportation services to users. The use of an Intelligent Transportation Systems (ITS) architecture encourages structured development and integration of individual systems [1] that leads to sharing of information and enables use and re-use of transportation information that is distributed across various independent systems. Services providing access to such transportation data might enable users to retrieve data ranging from information on places of interest, to prevailing road weather conditions, to expected journey times, to up-to-date public transport information. Such services might also enable suitably privileged users to interact with the infrastructure, for example, to request a change to a traffic light or to reserve a parking space.

The basis for the delivery of such context-aware services and information to users will be a conceptual abstraction for accessing and correlating the information associated with an ITS architecture. This paper presents a spatial programming model designed to provide a standardized way to build value-added transportation user services that deliver contextual transportation information derived from independent systems to users. The spatial programming supports a topographical location model and provides access to distributed context information based on potentially overlapping spatial and temporal aspects.

This enables services to exploit and act upon information from a variety of deployed (and novel) systems and services as well as to share information between them. The spatial programming model hides the complexity and diversity of the underlying systems and their data sources and provides services with a common view on the available information and its context. For example, a service might use the spatial programming model to retrieve public transport information, which might be provided by some underlying system, and then access relevant weather information provided by another system using the temporal and spatial context of this information.

The spatial programming model is part of the iTransIT ITS framework for integrating individual transportation systems and related services. The iTransIT framework has been motivated by the needs of Dublin City and its multi-layered distributed architecture has been designed to enable information integration and sharing across independent ITS and context-aware user services. The iTransIT framework has been developed in cooperation with the Traffic Office of Dublin City Council (DCC) in the Republic of Ireland. Detailed framework (and spatial programming model) requirements were informed by a comprehensive audit of existing and planned future intelligent transportation systems in the Dublin City area.

The proposed spatial programming model has been realized as part of a proof-of-concept architecture and data model that captures a variety of real transportation information derived from systems currently deployed in Dublin City. This implementation has been evaluated by building a user service for multi-modal urban journey planning that exploits information generated by a variety of underlying heterogeneous systems in a context-aware manner. The evaluation is based on transportation information relevant to and derived from a real urban environment and demonstrates how our programming model supports user service development and eventually enables user access to context information.

The remainder of this paper is structured as follows: Section II introduces the iTransIT framework for integrating individual transportation systems and related user services. Section III describes the spatial programming model and section IV outlines how the spatial programming model supports transportation user services. Section V presents our evaluation of this work outlining how the programming model provides city-wide access to the context information required by a multi-modal traveler information system. Finally, section VI concludes this paper by summarizing our work.

Manuscript received March 20, 2006. The work described in this paper was partially supported by the Dublin City Council in Ireland.

René Meier, Anthony Harrington, and Vinny Cahill are with the Distributed Systems Group in the Department of Computer Science at Trinity College Dublin, Ireland (phone: +353 1 608 1261; fax: +353 1 677 2204; e-mail: {rene.meier, anthony.harrington, vinny.cahill}@cs.tcd.ie).

1-4244-0094-5/06/$20.00 ©2006 IEEE

II. The iTransIT Architecture

As illustrated in Fig. 1, the iTransIT architecture structures legacy systems, iTransIT systems, and context-aware end-user applications into three tiers. These tiers define the relationships between systems and applications and provide a scalable approach for integrating systems and their context information as individual components can be added to a specific tier without direct consequences to the components in the remaining tiers. The relationships between systems and applications can be characterized according to the interaction paradigms that describe the possible information flows between legacy and iTransIT systems.

A. Architecture Tiers

The legacy tier provides for the integration of legacy systems and describes existing as well as future transportation systems that have not been developed to conform to the iTransIT system architecture and layered data model. Such legacy systems often feature a form of persistent data storage and might include systems for traffic and motorway management that have commonly been deployed in many urban environments.

The purpose of the iTransIT tier is to integrate transportation systems that model spatial information and implement the spatial application programming interface. This tier therefore comprises a federation of transportation systems that implement the spatial data model. The data model is distributed across these iTransIT systems, with each system implementing the subset of the overall model that is relevant to its operation. iTransIT systems maintain their individual information, which is often gathered by sensors or provided to actuators, by populating the relevant part of the spatial data model. However, some of the information maintained in an iTransIT system specific part of the data model may actually be provided by underlying legacy systems. Most significantly, traffic information captured in this tier is maintained with its temporal and spatial context; persistently stored data is geo-coded typically by systems exploiting a database with spatial extensions.

The systems that may exist in the iTransIT tier can be classified according to the paradigms they exploit when interacting with other legacy or iTransIT systems. Such iTransIT systems may be purpose built and therefore optimized to accommodate application or user-specific requirements or may be general purpose. As shown in Fig. 1, the framework may incorporate a general-purpose iTransIT Management system. The iTransIT Management system is the canonical application of this domain and is expected to implement a major part of the spatial data model. It typically serves as a main repository for geo-coded data generated and used by connected legacy and iTransIT systems.

The application tier includes value added services that provide context-aware user access to and interaction with traffic information. These services use the distributed data model and the associated context to access information potentially provided by multiple systems and might include a wide range of interactive (Internet-based) and embedded control services ranging from monitoring of live and historical traffic information to the display of road network maps.

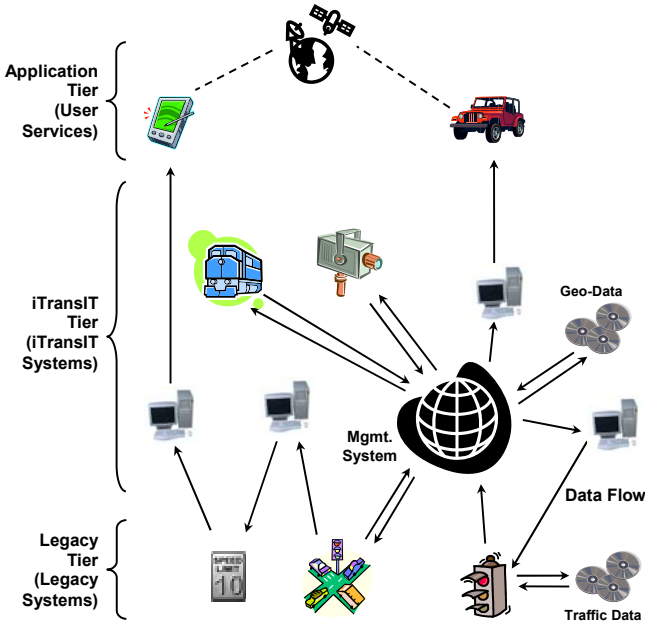

Fig. 1. iTransIT ITS framework overview.

B. Common Spatial Data Model

The spatial data model, common to all iTransIT systems, is comprised of a set of potentially distributed layers [2] and represents the central component of these systems. Individual iTransIT systems implement one or more of these layers (or parts of layers) and maintain the static, dynamic, live, or historical traffic data available in a particular layer. For example, a system might implement a data layer describing the current weather conditions while another layer capturing intersection-based traffic volumes might be maintained by a different system.

Some of the information captured in data model layers may be generated or used by legacy systems. Such information is mapped to a legacy system through data flows. These flows can be described using a set of flow classes, including event, stream, request/response, configuration and alarm flows, based on the characteristics and requirements of communication links provided by the KAREN framework architecture [3]. Using these descriptions, individual iTransIT systems implement interfaces that map specific legacy data to their data layers. This approach enables the use of communication technologies that can address the requirements of particular systems and their respective data flows. The objective of an iTransIT system might be to handle a certain data subset efficiently and to provide specific guarantees for the

delivery of the data. For example, an iTransIT system may employ real-time communication technology to connect to a legacy system that is capable of supporting strong delivery guarantees.

III. The Spatial Application Programming Model

The spatial programming model provides a standardized way for user services to access and use information and context that is distributed across independent systems and related services. The spatial programming model provides common access to such distributed information based on overlapping context thereby enabling services (and users) to exploit and act upon information from a variety of systems.

A. Abstracting Information and Context

The spatial programming model uses a small set of predefined types for composing information and context, in which context is any information that can be used to characterize the situation of an information element [4], to ensure interoperability between data sets captured across distributed systems. These types are used to model data sets and their context according to the different roles data sets can assume as *spatial objects*. Spatial objects represent information as a series of parameters and context as attributes. Such types are central to providing user services with a common view on the wide range of information and the associated context that might be available in an ITS infrastructure. They hide the complexity and diversity of the independent systems and data sources comprising such infrastructures and represent the hooks for information integration through overlapping context such as space and time.

Developing such types is non trivial for any programming model for significant systems and is especially complex for ITS infrastructures due to the scale and multitude of inter-relationships that exist between sensors, systems, services, users, and their data sets. Lehman et al. [5] suggest an exhaustive ontology for defining how context information can be shared between applications in augmented areas. However, based on our experience with the transportation domain, we have found that a relatively small number of types suffices to decompose a domain model. Using a small set of (coarse-grain) types rather than attempting to model the entire world in detail simplifies management and maintenance in light of continuously evolving infrastructures. Novel systems or services are expected to be modeled using combinations of existing types whereas an exhaustive model might have to be expanded to capture the specific characteristics of novel systems.

The types for modeling information and context as spatial objects currently supported by the spatial programming model are summarized in Fig. 2. They have been designed as a series of abstract object types and include three main types for modeling global information, which are *real world, system and data object*, as well as types for modeling context.

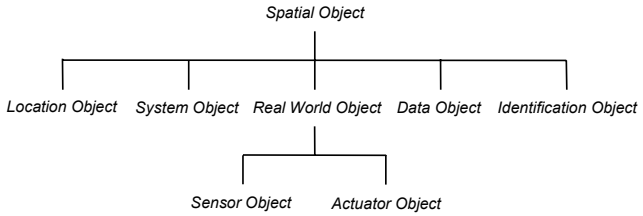

Fig. 2. Information and context abstractions.

The three information types model the different roles that objects can assume within the spatial programming model. System objects represent general information describing software components, including systems and services, while real world objects represent physical entities. For example, system objects might capture operational status from a car parking system or from a journey time estimation service whereas real world objects might model roads and junctions. Sensor and actuator objects are specializations of real world objects and are used for modeling explicit infrastructural entities for example, detector loops and variable message signs of a car parking system. Data objects model any static or dynamic information from systems or services and might be used to model car parking opening times and rates charged. Based on an audit of deployed (and planned) transportation systems and services in the Dublin City area [6], we found that these categories of information types are sufficient to cover possible data sets. Novel information can be integrated using spatial objects composing sets of parameters that model such data sets.

The main context type of the spatial programming model is the *location object*. Location objects are based on a topographical location model that uses geometry to model the space occupied or covered by an infrastructural element, a system or a service. The spatial programming model also supports temporal context. Temporal context is modeled implicitly, i.e., incorporated in other information types, rather than explicitly as a specific object. This enables information objects to include date and time attributes for representing their temporal context such as creation time and temporal validity. And finally, *identification objects* provide a type for logical identity, for example, to identify the name of a system or a service.

B. Modelling Space

The spatial programming model supports a topographical approach to modelling space. The relevant spatial context of sensors, systems, services and even users is modelled as a geometric shape. Individual shapes are defined by a sequence of coordinates based on a chosen, well-known coordinate system. These shapes explicitly represent spatial context derived form the real world. They may reflect the physical appearances of spatial objects modelling occupied space or may describe areas of interest that specify the regions covered by services. For example, a city-wide car parking system might use the spatial model to define the physical locations occupied by its car parks whereas a road

weather service might use the spatial model to outline the locations occupied by weather stations as well as the areas to which reports from individual stations apply.

Using a topographical approach to modeling space enables systems, services, and applications to independently define and use potentially overlapping spatial context in a consistent manner. Unlike topological approaches [7], in which geographical relationships between spatial objects are described explicitly, topographical models define relationships between spatial objects implicitly and without explicit interactions between objects. The relations between spatial objects (and ultimately systems and users) are defined by the position of their respective shape within the common coordinate system. This is particularly significant where multitudes of independent systems are distributed over large geographical areas and direct communication across systems may be limited or expensive. Applications using the spatial model can exploit these implicit relations to link diverse information together for a user specific purpose. They may access spatially related information for example, by means of exploiting the distance between shapes or by exploiting containment and intersection relations. This might for example enable a vehicle-based information system to retrieve the exact locations of car parking facilities within a certain distance from its current location.

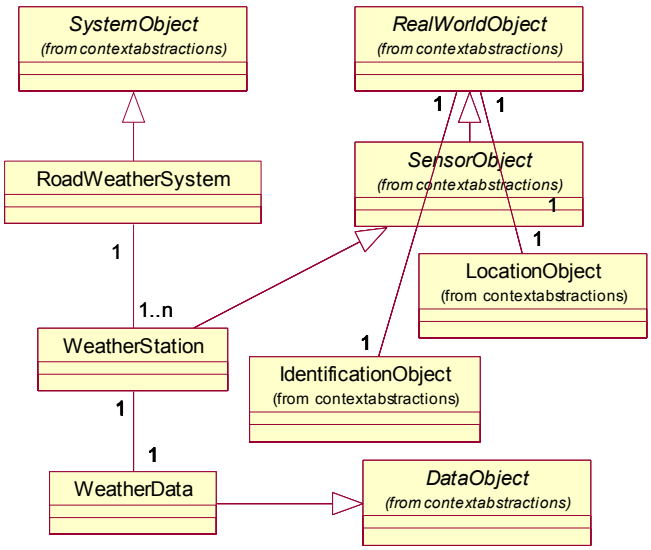

Fig. 3. Modeling a road weather system.

The spatial programming model supports the model for defining geometric shapes defined by the OpenGIS standard [8]. Spatial objects can be represented by geometry types ranging from a point, to a line, to a polygon, to combinations of polygons. Points might be used to define the location of a specific traffic signal or an individual user. Individual polygons might represent the spatial context of a car park or an area of interest whereas a series of (overlapping) polygons might be used to compose a spatial model of a transportation network comprising roads, lanes, and intersections.

As mentioned above, these geometric shapes are specified using a common coordinate system. The selection of such a system depends on the domain for which the spatial programming model is being realized. Coordinates derived from third party location sensors, such as Global Positioning System (GPS) receivers, are mapped onto the chosen reference system if they are based on another system. For example, GPS coordinates may need be converted into a regional reference system chosen for a specific space. The Irish national grid reference system, a system of geographic grid references commonly used in Ireland, has been chosen as the coordinate system in our prototype.

C. *Modeling Data*

The spatial programming model defines a set of types for modelling the different roles spatial objects (and the context information they represent) can assume. Systems and services model their data using these types and a particular system may use and combine several types to accurately capture the roles of individual data sets. The example shown in Fig. 3, illustrates how a road weather system might use a system object to model general system data and a set of sensor objects to model individual weather stations. Each weather station comprises a location and an identification object and includes a data object that captures the actual measurements.

Spatial objects must specialize at least one of our types for modeling information and context. However, depending on their role, they may derive from several types. Table 1 summarizes how these types can be combined outlining the semantics for composing information and context into spatial objects. As outlined in the real world object row, Table 1 shows that a real world object must comprise a location and an identification object and that it may include a set of data objects and a set of other real world objects. The compulsory containment of a location object is a reflection of the fact that real world objects are expected to model the physical space they occupy. In contrast, system and data objects may or may not comprise a location object and such a location object is probably modeling the space to which a system's or data object's information applies. Note that sensor and actuator objects are specializations of real world objects that share the same composition semantics.

Table 1. The semantics for composing information and context types.

	System Object	**Real World (Sensor, Actuator) Object**	**Data Object**	**Loc. Object**	**Ident. Object**
System Object	0..n	0..n	0..n	0..1	0..1
Real World (Sensor, Actuator) Object	0	0..n	0..n	1	1
Data Object	0	0	0..n	0..1	0..1

D. *Using the Spatial Model*

Systems use spatial objects to model their contextual

information and implement the spatial application programming interface to provide pervasive access to these objects. Each system models the subset of the spatial objects that is relevant to its respective purpose and context-aware services exploit the spatial application programming interface to integrate and share information in a common way regardless of the specifics of the system implementing a particular part of the spatial model.

As shown below, the operations of the spatial application programming interface provide a means for services to manage, locate and access spatial objects. A set of operations is available for locating spatial objects using geometric queries or queries based on parameters of objects. Geometric queries are based on a geometry class that defines OpenGIS shapes including points and polygons. Parameter-based queries use the container class outlined below to describe the parameter and attribute values of spatial objects. The parameter class includes native data values and may include the relevant temporal attributes of data objects. This class can be used in connections with queries but may also be used to access the typed parameter and attribute values of spatial objects. The spatial application programming interface enables services to locate spatial objects using a variety of queries ranging from selection based on a parameter value, to selection based on temporal context, to selection based on spatial context, to combinations of these. For example, a weather station may be selected using the value of a measurement, the temporal occurrence of a measurement or the location of the station. Such queries may identify zero, one or more objects. For example, selecting the bus stops of a certain bus route in a particular area might identify multiple suitable stops. Spatial objects are uniquely identified within a given system by a type and identifier pair. These pairs are typically the result of some selection operation and may be used to either retrieve or update the parameters of spatial objects. A service might use bus stop and identifier pairs to retrieve the addresses and timetables of previously located stops.

Significantly, the spatial programming model enables a federation of independent systems to model their respective information and context *locally* as spatial objects. Each of these systems implements the spatial application programming interface to provide access to its respective set of spatial objects. This enables services to use, share, locate and correlate these distributed objects using a common set of context operations irrespective of the complexities of the systems accommodating the objects and without the need for an overall close integration of the systems. This mapping of the spatial model and its programming interface onto individual systems therefore provides for context-aware services in large-scale and heterogeneous environments.

```
interface S_API {
  void insert(String elementType,
              OrderedParameterValues parValues);
  void remove(String elementType, int id);
  int[] select(String elementType, Geometry loc);
  int[] select(String elementType, String parName,
               Parameter parValue);
  int[] select(String elementType, Geometry loc,
               String parName, Parameter parValue);
  int[] select(String elementType);
  ElementTypeAndId[] select(Geometry loc);
  Geometry select(String elementType, int id);
  void update(String elementType, int id,
              String parName[],
              Parameter parValues[]);
  Parameter[] retrieve(String elementType, int id,
                       String parName[]);
}

class Parameter{
  Calendar creationDate;
  Calendar modificationDate;
  Long retrievalLatency;
  Long expectedLifetime;
  Double confidenceLevel;
  String parameterValue;

  Integer getIntegerParameterValue();
  Double getDoubleParameterValue();
  String getStringParameterValue();
  Calendar getDateParameterValue();
…
}
```

IV. Transportation User Services

The spatial application programming interface exposes the layered data model to transportation user services enabling them to access the spatial information that is relevant to them. Fig. 4 illustrates that such user services capture the content to be made assessable to the user. This service content may be derived directly from the spatial programming interface or may be generated by the service. Directly derived content represents transportation information provided by either iTransIT systems or legacy systems. Such content might include prevailing road congestion levels and public transport information such as train timetables and bus locations. Content that is generated by the service represents the actual value added. The stimuli for generating this additional content are usually derived, via spatial programming model, from the underlying systems as well. For example, a traveler information service might use information on public transportation in combination with road network information to plan and route user journeys between points of interest.

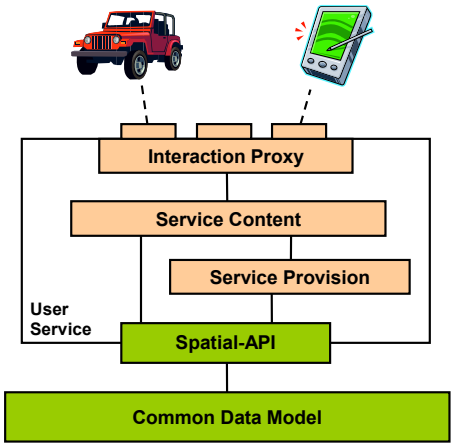

Fig. 4. Building user services.

The content of a service can be delivered to users through a proxy handling interactions and sessions. Remote access to

such a service specific interface may be enabled through widely used communication technologies for example, based on CORBA, Web Services, or General Packet Radio Services (GPRS). Naturally, a specific service may support multiple (diverse) communication technologies for user interaction enabling users to access, request, and indeed provide information using a range of means. For example, a user might request journey information using either a handheld device with a wireless connection or a messaging service on a mobile phone.

V. ASSESSMENT

This section evaluates the spatial programming model for context-aware user services proposed in this paper. The main objective of the experiments has been to assess the feasibility of our programming model providing access to information generated by a variety of heterogeneous systems in a context-aware manner. The assessed transportation service scenario demonstrates that our programming model enables service and eventually user access to context information derived from a real urban environment through correlation of overlapping spatial context. This evaluation therefore demonstrates that using a spatial programming model enables the delivery of context-aware services and information to users.

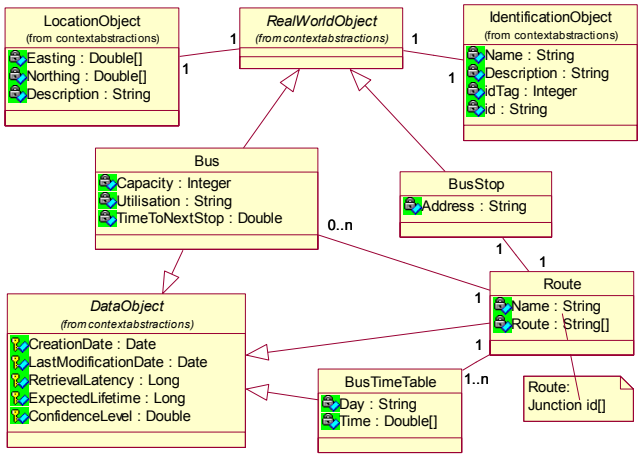

Fig. 5. Spatial objects modeling public transport information.

The scenario has been derived from the requirements of a smart traveler information service enabling travelers to plan journeys involving multiple forms of transportation including walking, public transport, cycling, and private vehicles thereby bridging the coordination gap between these modes of transportation by suggesting journey routes according to traveler preference and availability of transportation means. The scenario has been assessed using a prototypical implementation of an iTransIT Management system as a platform for transportation services. This Management system implements the spatial application programming interface and uses spatial objects to model information concerning a range of transportation systems currently deployed in Dublin City. The system includes data layers modeling the road network comprising intersections, roads, lanes, traffic counts, traffic volumes, and congestions levels as well as the public transport network consisting of bus routes, stops, lanes, timetables and bus locations. It also includes data layers modeling parking information and road weather data. These layers integrate data provided by a range of real legacy systems including the main traffic management system, a public transport information service, a congestion level application, a road weather service and a car parking information system. Fig. 5 shows a small set of the spatial objects modeling these layers that have been implemented as relational tables in a MySQL database with spatial extension. The information from these spatial objects has been provided by the traffic management system, the public transport information service and by a journey time monitoring system.

A. The Evaluation Scenario

The evaluation scenario includes a tourist using the context-aware traveller information service to locate public transport stations within walking distance of her current location. The tourist has just visited The Book of Kells museum at Trinity College Dublin and is about to leave campus through the Nassau Street gate. She remembers that she used the number 15 bus to travel from her hotel to the city centre and would therefore like to locate nearby bus stops of this route.

She uses a handheld device with wireless service access to enter her query into the traveler information service, providing bus route number 15 and 5 minutes walking distance from her current location as parameters. The service uses coordinates derived from its GPS receiver (converted into Irish national grid coordinates) and an average pedestrian pace of 1.36m/s [9] to define the geometric shape of the search area. The service then uses the spatial application programming interface as outlined below to access the relevant context information.

```
1 int[] busStopId = sapi.select("BusStop",
                                  searchArea);
  for (int i = 0; i < busStopId.length; i++) {
2   Parameter busStopName=sapi.retrieve("BusStop",
                                busStopId[i],"Name");
3   Geometry busStopLocation =
            sapi.select("BusStop", busStopId[i]);
4   Parameter linkToRoute =
            sapi.retrieve("BusStop", busStopId[i],
                          "route_autoId");
    int routeId = linkToRoute.getIntegerValue();
5   Parameter routeName = sapi.retrieve("Route",
                                routeId, "Name");
6   if (routeName.getStringValue().equals("15-
outbound")) ||
       (routeName.getStringValue().equals("15-
inbound")) {
7     //use results
    }
  }
```

The service might use a geometric query to locate all spatial objects representing bus stops in the give search area (1) and retrieve the parameters and attributes of these objects that describe the names and locations of specific bus stops (2, 3). The service then proceeds to identify the spatial objects that describe the routes associated with these bus

stops. These "links" to route objects are modeled as parameters that can be retrieved from bus stop objects (4). They are subsequently used to retrieve the names of the bus stop routes (5) and information related to the previously indicated bus route (6) can then be used to advise the user (7). The results of such a scenario for locating bus stops within walking distance can be found in Table 2. Bus stops for both city centre-bound and suburb-bound stops have been retrieved since the user did not specify her preferences. Naturally, a traveler information service would display this information as an overlay to a map of Dublin City rather than in table form. Such an overlay might include the bus stop names and the headings of buses. This might further assist the user in locating and eventually walking to a convenient bus stop.

Table 2. Locating public transport stations within walking distance.

Bus Stop Name	**Route Name**	**Bus Stop Location (Irish national grid coordinates)**
Kildare Street	15-outbound	(316230.8575, 233593.6385)
Dawson Street Upper	15-inbound	(316063.4310, 233792.1260)
Dawson Street Lower	15-inbound	(316036.3947, 233612.0083)
Suffolk Street	15-inbound	(315924.9190, 233981.6965)
Nassau Street	15-outbound	(316202.2930, 233883.7390)
College Green	15-outbound	(316038.3422, 234186.3123)

This scenario demonstrates how a context-aware user service might use the spatial programming model to locate real-world entities in a given area of interest and how it might exploit explicit associations between spatial objects. Similar queries can be used by a range of related scenarios. For example, after selecting a bus stop, the user might wish to see the relevant timetable for the next hour or might wish to use the address of her hotel to locate a convenient stop near her destination and to display the route the bus will take. Other related scenarios might include retrieving the congestion levels along the route in order to get an indication of whether the bus is likely to be on time. Such a scenario might also be of interest to someone traveling by car to the airport or to work. These related scenarios have been implemented but due to space limitations are not describe in further detail.

This assessment is based on scenarios that access information integrated in the spatial model through a single spatial application programming interface. However, a context-aware user service may concurrently use multiple spatial application programming interfaces to access spatial objects in a similar way. The overlapping context of such distributed spatial objects may be used similarly to correlate objects. For example, the location of a bus stop available from one spatial application programming interface might be used to locate nearby train stations through another interface.

VI. CONCLUSIONS

This paper presented a programming model for building value-added transportation services that deliver contextual information derived from independent systems to users. The spatial programming model uses a small set of predefined types to model distributed context information as spatial objects. This provides a common view on such information and enables services and ultimately users to exploit, act upon and share information based on overlapping spatial and temporal aspects. The spatial programming model supports a topographical location model in which spatial context derived form the real world is explicitly represented by shapes that reflect occupied space or describe areas of interest. This enables systems distributed over large geographical areas to independently define and use spatial context in a consistent manner.

The spatial programming model is part of the iTransIT framework for integrating individual transportation systems and services that has been motivated by the needs of Dublin City. The multi-layered distributed iTransIT architecture supports a distributed data model in which individual systems maintain one or more layers of the overall data model.

The evaluation of the spatial programming model has been based on a prototypical implementation of an iTransIT management system that uses spatial objects to model real information relevant to and derived from a range of transportation systems currently deployed in Dublin City. The assessed transportation service scenario demonstrated that our programming model enables service and eventually user access to context information derived from a real urban environment through correlation of overlapping spatial context. This evaluation therefore demonstrated that using a spatial programming model enables the delivery of context-aware transportation services and information to users.

A more substantial iTransIT prototype is currently being realized incorporating a considerable number of diverse systems and transportation data sets. This prototype will enable a further evaluation of the spatial programming model using a wide variety of transportation service scenarios that is expected to produce significant results. Furthermore, Dublin City Council is concurrently evaluating the iTransIT framework based on its application to real transportation systems planned for Dublin City.

REFERENCES

[1] J. McQueen and B. McQueen, *Intelligent Transportation Systems Architectures*. Boston, USA: Artech House Books, 1999.

[2] R. Meier, A. Harrington, and V. Cahill, "A Framework for Integrating Existing and Novel Intelligent Transportation Systems," in *Proceedings of the 8th International IEEE Conference on Intelligent Transportation Systems (IEEE ITSC'05)*. Vienna, Austria: IEEE Computer Society, 2005, pp. 650-655.

[3] R. A. P. Bossom, "European ITS Framework Architecture - Communication Architecture, Annex 1: Supporting Information for Communications Analysis," vol. D3.3: European Communities, 2000.

[4] A. Dey and G. Abowd, "Towards a Better Understanding of Context and Context-Awareness," in *Workshop on The What, Who, Where, When, and How of Context-Awareness, as part of the 2000 Conference on Human Factors in Computing Systems (CHI 2000)*. The Hague, The Netherlands, 2000.

[5] O. Lehmann, M. Bauer, C. Becker, and D. Nicklas, "From Home to World - Supporting Context-aware Applications through World Models," in *Proceedings of Second IEEE International Conference on Pervasive Computing and Communications (Percom'04)*. Orlando, Florida: IEEE Computer Society, 2004, pp. 297-308.

[6] R. Meier, A. Harrington, and V. Cahill, "Audit of ITS Applications and Services in Dublin City," Trinity College, Dublin, Ireland, Dublin City Council iTransIT Deliverable, August 2004.

[7] M. Bauer, C. Becker, and K. Rothermel, "Location Models from the Perspective of Context-Aware Applications and Mobile Ad Hoc Networks," *Personal and Ubiquitous Computing*, vol. 6, pp. 322-328, 2002.

[8] Open GIS Consortium Inc, "OpenGIS Simple Features Specification for SQL, Revision 1.1," OpenGIS Project Document 99-049, 1999.

[9] T. F. Fugger, B. C. Randles, A. C. Stein, W. C. Whiting, and B. Gallagher, "Analysis of Pedestrian Gait and Perception–Reaction at Signal-Controlled Crosswalk Intersections," National Research Council, Washington, D.C, USA, Transportation Research Record 1705 TRB 00-1439, 2000.