A Spatial Programming Model for Real Global Smart Space Applications

René Meier, Anthony Harrington, Thomas Termin, and Vinny Cahill

Distributed Systems Group, Department of Computer Science
Trinity College Dublin, Ireland
{rene.meier, anthony.harrington, thomas.termin,
vinny.cahill}@cs.tcd.ie

Abstract. Global smart spaces are intended to provide their inhabitants with context-aware access to pervasive services and information relevant to large geographical areas. Transportation is one obvious domain for such global smart spaces since applications can be built to exploit the variety of sensor-rich systems that have been deployed to support urban traffic control and highway management as well as within individual vehicles. This paper presents a spatial programming model designed to provide a standardised way to build context-aware global smart space applications using information that is distributed across *independent* (legacy, sensor-enabled, and embedded) systems by exploiting the overlapping spatial and temporal attributes of the information maintained by these systems. The spatial programming model is based on a *topographical approach* to modelling space that enables systems to indepe-ndently define and use potentially overlapping spatial context in a consistent manner and in contrast to topological approaches, in which geogra-phical relation-nships between objects are described explicitly. Moreover, this approach facilitates the *incremental* construction of global smart spaces since the underlying systems to be incorporated are largely decoupled. The programming model has been evaluated by building a context-aware service for multi-modal urban journey planning, as part of the development of an overall architecture for intelligent transportation systems in Dublin.

1 Introduction

Global smart spaces extend the vision of pervasive computing, in which everyday objects communicate and collaborate to provide information and services to users, to large geographical areas [1]. They extend the notion of objects cooperating in a home or an office to the level of towns, cities, and even countries by integrating a variety of sensor-based and other systems to provide truly pervasive context-aware services. Such global smart environments will be heterogeneous as they likely will comprise a multitude of sensors, networks, and ultimately systems. They will provide access to information and services ranging from pervasive access to personal and professional information, to city-wide information systems [2, 3], to context-aware traveller assistance [4, 5], to optimised urban traffic control [6]. Users moving in such sensor-augmented spaces may use handheld devices, such as mobile phones and Personal Digital Assistants (PDAs), or integrated devices, such as (vehicular) on-board

F. Eliassen and A. Montresor (Eds.): DAIS 2006, LNCS 4025, pp. 16 – 31, 2006.

computers, to interact with these spaces and to use the services that they provide. Embedded control systems may likewise exploit these spaces to offer context-aware urban traffic control, such as public service vehicle priority.

Global smart spaces are on the verge of becoming a reality in the transportation domain where very many heterogeneous sensor-rich systems have already been deployed in towns and cities and along national road networks. Such a global smart space might enable users to access information ranging from information on places of interest, to prevailing road and weather conditions, to expected journey times, to up-to-date public transport information. It 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.

Programming Global Smart Space Applications. The basis for the provision of context-aware services and information to users will be the integration of the individual systems associated with global smart spaces into comprehensive platforms. This paper presents a programming model designed to provide a standardised way for global smart space applications to access context information that is provided by *independent* systems and related services. The spatial programming model supports a *topographical* location model and provides access to distributed context information based on (overlapping) temporal and spatial aspects. This enables applications 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 applications 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 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 Intelligent Transportation Systems (ITS) and pervasive context-aware user services. It enables *incremental* integration of independent systems and services over time while minimising the impact of such expansion as changes are local to the new system. This software architecture for global smart spaces proposes a layered data model to facilitate data exchange between systems and services with diverse data sets, quality of service requirements, and functional organizations. Data layers are defined within a common context model along the dimensions of space and time and may be distributed across multiple systems. Individual systems maintain one or more layers of the overall data model. This distribution of layers across a series of systems effectively allows applications to access elements of a certain part of the model with a specific quality of service. For example, a data layer might provide video streams from traffic cameras while another layer might maintain city-wide parking information provided by a car parking system. Applications may use the spatial programming model to access either or both of these layers with the quality of service of the respective information. This scenario also illustrates that systems may be integrated gradually and with minimal impact on other systems. Each of these layers

might be integrated at a different time and the integration of one layer does not affect the data captured in the other layer. An application using the spatial programming model to access information from the video layer might eventually be updated to access the car parking layer as well. 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.

Realising Global Smart Space Applications. The proposed spatial programming model has been implemented 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 programming model implementation has been evaluated by building a pervasive service for multi-modal urban journey planning. Such a smart traveller information service can be considered a canonical global smart spaces application since it 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 enables application and eventually user access to such pervasive context information. In general, it is expected that the increased availability of re-usable information from a variety of independent systems will enable higher-level policies to be translated more easily into real world actions and will facilitate the emergence of novel transportation applications and truly pervasive context-aware user services.

Organisation of This Paper. The remainder of this paper is structured as follows: Section 2 surveys related work. Section 3 presents the spatial programming model and section 4 describes how this programming model has been realised as part of a framework for integrating independent transportation systems. Section 5 presents our evaluation of this work outlining how the spatial programming model provides global access to the context information required by a multi-modal traveller information system. Finally, section 6 concludes this paper by summarising our work.

2 Related Work

Temporal, spatial and quality of service attributes represent types of meta-data that may be integrated into a context model to provide more intelligent and focused use of data [7]. This approach has been applied in the Nexus framework [8] which provides a common context model infused with spatial information to build world models that are distributed across spaces possessing rich context data sources, known as Augmented Areas. The context model is presented as a global object-based ontology for developing interoperable world models. This interoperability is ensured through the use of a common but large data schema, the Standard Class Schema, to define various world models. The authors have defined a simple spatial query language that can be used to interact with objects representing an Augmented Area. An interface known as an Augmented World model provides a federated global view on all compliant local models. The focus of our work has been to develop a more

constrained yet expressive set of abstractions which are used to both facilitate data modelling and to provide the basis for our spatial application programming interface. Using such a constrained set of abstractions simplifies management and maintenance in light of continuously evolving global smart spaces as novel systems are expected to use combinations of existing abstractions.

Gaia [9] is a canonical example of a middleware infrastructure to enable active or smart spaces in ubiquitous computing habitats that emphasises the notion of space programmability. Gaia extends the notion of traditional operating systems to ubiquitous computing environments by providing components such as the Context File System and an event manager to track active space state information. Gaia focuses on managing resources contained in physical spaces. User data and applications are abstracted into a user virtual space and can be mapped dynamically to the resources located in the current environment. Applications developed for a Gaia active space use a comprehensive set of services at runtime. The iTransIT framework adopts a different approach in that it uses a set of context abstractions exposed through the spatial programming model to provide an interface to a global smart space populated by heterogeneous systems. Aside from calls to the spatial application programming interface, systems may operate independently of the iTransIT framework.

Smart Messages [10] is a lightweight architecture similar to mobile agents that aims to make Space a first-order programming construct and describes a space-aware programming model for outdoor distributed embedded systems called Spatial Programming. In this model, content or services provided by nodes are accessed using spatial references. These are defined as {space:tag} pairs that are mapped to systems embedded in the physical space. These spatial references are used by various applications to transparently access network resources in a similar fashion to physical memory access using variable names in conventional systems. Our approach to accessing information in a global smart space is more generic compared to this {space:tag}-based naming scheme in that information can be located using multiple context dimensions including space and time as well as any functional aspect of the information. Information can be shared and integrated by exploiting combinations of these aspects and by exploiting overlapping context.

3 The Spatial Application Programming Model

The spatial programming model provides a standardised way for global smart space applications 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 applications to exploit and act upon information from a variety of systems and services as well as to share information between them.

3.1 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

characterise the situation of an information element [11], 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 in a global smart space as *spatial objects*. Spatial objects represent information as a series of parameters and context as attributes. Such types are central to providing applications with a common view on the wide range of information and the associated context that might be available in a global smart space. They hide the complexity and diversity of the independent systems and data sources comprising global spaces 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 global smart spaces due to the scale and multitude of inter-relationships that exist between sensors, systems, services, users, and their data sets. Lehman et al. [8] suggest an exhaustive ontology for defining how context information can be shared between applications in augmented areas. However, based on our experience with a real global smart space in the transportation domain, we have found that a relatively small number of types suffices to decompose a global smart space 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 spaces. Novel systems or services are expected to be modelled 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 modelling information and context as spatial objects currently supported by the spatial programming model are summarized in **Fig. 1**. They have been designed as a series of abstract object types and include three main types for modelling global information, which are *real world, system and data object*, as well as types for modelling context.

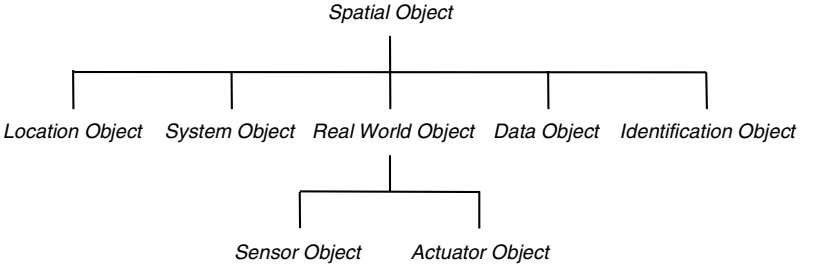

Fig. 1. Information and context abstractions

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. In a transportation smart space 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 specialisations of real world objects and are used for modelling 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 [12], we found that these categories of information types are sufficient to cover possible data sets in such a global smart space. 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 modelled 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.

3.2 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 modelling space enables systems, services, and applications to independently define and use potentially overlapping spatial context in a consistent manner. Unlike topological approaches [13], 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 in global smart spaces 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.

The spatial programming model supports the model for defining geometric shapes defined by the OpenGIS standard [14]. 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 of the global smart space for which the spatial programming model is being realised. 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.

3.3 Modelling 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 within a global smart space. 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. 2**, 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 specialise at least one of our types for modelling information and context. However, depending on their role, they may derive from several types. **Table 1** summarises 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.

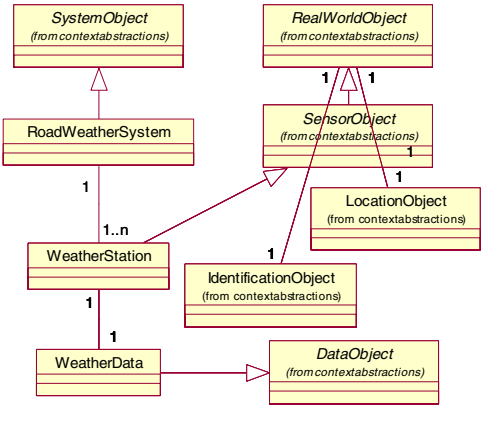

Fig. 2. Modeling a road weather system

In contrast, system and data objects may or may not comprise a location object and such a location object is probably modelling the space to which a system's or data object's information applies. Note that sensor and actuator objects are specialisations 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**	**Location Object**	**Identification 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

3.4 Modelling Temporal Context

In addition to supporting spatial context, the spatial programming model also supports context along the dimension of time. The temporal relations between spatial objects are defined by a set of attributes. This set of attributes has been derived from our study of the transportation infrastructure in Dublin City [12] and are summarised in **Table 2**. The data object type includes these attributes and spatial objects model their temporal context by deriving from this type. Data objects also include a *ConfidenceLevel* attribute for modelling the accuracy of the captured data.

Table 2. Temporal context attributes of data object types

Attribute Name	**Description**
CreationDate	Time of data object creation
LastModificationDate	Time the data object was last updated
RetrievalLatency	Expected latency for retrieving the captured data
ExpectedLifetime	Expected duration to the next data object update
ConfidenceLevel	Level of confidence in the accuracy of the captured data

Applications may exploit temporal relations between spatial objects in the same way as they exploit spatial relations to link diverse information together for a user-specific purpose. They may access temporally related information, for example, by means of correlating modification time. Significantly, applications may exploit context along a combination of the spatial and temporal dimension. This might enable a road-user information system to use the location and time of an accident to retrieve the prevailing weather conditions at the accident site and subsequently to advice drivers of similarly dangerous road conditions.

3.5 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 applications 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 applications 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 applications 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. An application 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 applications 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 truly pervasive context-aware applications and services in global 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();
…
}
```

4 The iTransIT Framework

The spatial programming model has been realised as part of a framework and a data model for integrating independent intelligent transportation systems. As illustrated in **Fig. 3**, 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.

4.1 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 objects and implement the spatial applica-tion 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

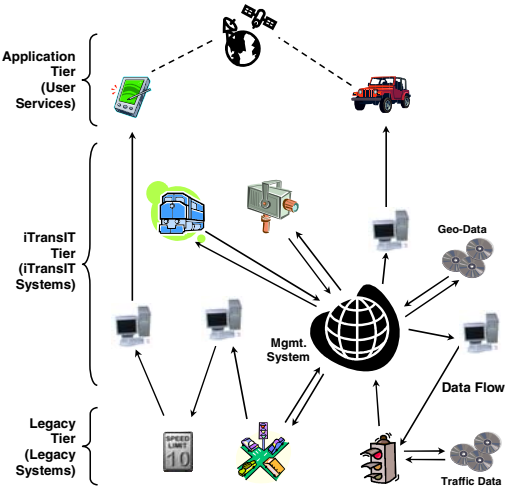

Fig. 3. iTransIT ITS architecture framework overview

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 extension.

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. 3**, 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.

4.2 Common Spatial Data Model

The spatial data model, common to all iTransIT systems, is comprised of a set of potentially distributed layers and represents the central component of these systems. As shown in **Fig. 4**, 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.

The spatial application programming interface exposes this layered data model to other iTransIT systems or indeed user services. Remote access to this interface may be enabled through widely used communication technologies and query languages based on CORBA and Web Services.

Some of the information captured in data model layers may be generated or used by legacy systems. Such information is mapped to a legacy

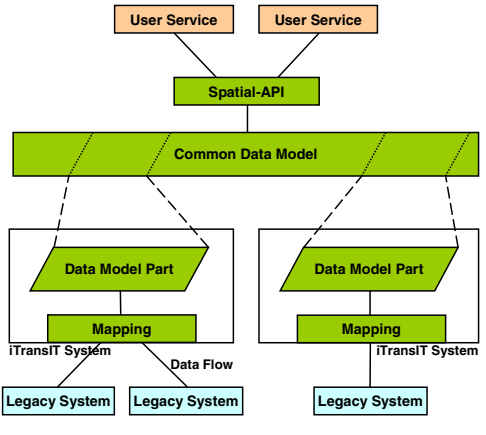

Fig. 4. iTransIT system architecture and common data model

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 [15]. 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.

5 Assessment

This section evaluates the spatial programming model for global smart space applications 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 application scenario demonstrates that our programming model enables application and eventually user access to pervasive 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 integration of individual systems associated with a global smart space into a comprehensive platform for the provision of context-aware services and information to users.

The application scenario has been derived from the requirements of a smart traveller information service enabling travellers 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 traveller preference and availability of transportation means. Such a service can be considered a canonical global smart spaces application since it exploits context information generated by a variety of independent systems. The scenario has been assessed using a prototypical implementation of an iTransIT Management system as a platform for pervasive services. This Management system implements the spatial application programming inte-rface and uses spatial objects to model information concerning

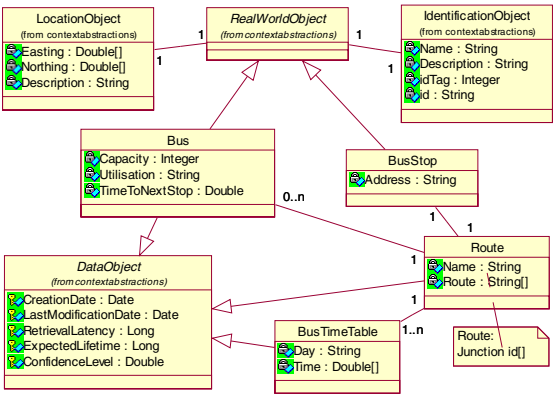

Fig. 5. Spatial objects modeling public transport information

a range of transportation systems currently deployed in Dublin City. The system includes global context layers modelling 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 system context layers modelling 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 modelling 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.

5.1 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 traveller 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 [16] 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 given 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 modelled 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 3**. Bus stops for both city centre-bound and suburb-bound stops have been retrieved since the user did not specify her preferences. Naturally, a traveller 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 3. 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 application 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 travelling 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.

6 Summary and Conclusions

This paper presented a programming model for global smart space applications to access context information provided by independent systems and related services. 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 applications to exploit, act upon and share information based on overlapping temporal and spatial 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 global smart spaces in the transportation domain that has been motivated by the needs of Dublin City. The multi-layered distributed iTransIT architecture enables incremental integration of independent systems and services over time while minimising the impact of such expansion as changes are local to the new system. The distributed data model, in which individual systems maintain one or more layers of the overall data model, facilitates data exchange between systems and services with diverse contextual data sets and functional organizations.

The evaluation of the spatial programming model is 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 scenario demonstrated that our programming model enables application and eventually user access to pervasive context information concerning a real urban environment through correlation of overlapping spatial context. This evaluation therefore demonstrates that using a spatial programming model enables the integration of individual systems associated with a global smart space into a comprehensive platform for the provision of truly pervasive context-aware services and information to users.

Acknowledgements. The work described in this paper was supported by the Dublin City Council in Ireland.

References

[1] A. Dearle, G. Kirby, R. Morrison, A. McCarthy, K. Mullen, Y. Yang, R. Connor, P. Welen, and A. Wilson, "Architectural Support for Global Smart Spaces," in *Proceedings of the 4th International Conference on Mobile Data Management (MDM 2003)*, *LNCS 2574*. Melbourne, Australia: Springer-Verlag, 2003, pp. 153-164.

[2] K. Cheverst, N. Davies, K. Mitchell, A. Friday, and C. Efstratiou, "Experiences of Developing and Deploying a Context-aware Tourist Guide: The GUIDE Project," in *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCom 2000)*. Boston, Massachusetts, USA: ACM Press, 2000, pp. 20-31.

[3] G. D. Abowd, C. G. Atkeson, J. Hong, S. Long, R. Kooper, and M. Pinkerton, "Cyberguide: A Mobile Context-Aware Tour Guide," *ACM Wireless Networks*, vol. 3, pp. 421-433, 1997.

[4] T. Sivaharan, G. Blair, A. Friday, M. Wu, H. Duran-Limon, P. Okanda, and C.-F. Sørensen, "Cooperating Sentient Vehicles for Next Generation Automobiles," presented at The First ACM International Workshop on Applications of Mobile Embedded Systems (WAMES'04), Boston, Massachusetts, USA, 2004.

[5] J. Kjeldskov, S. Howard, J. Murphy, J. Carroll, F. Vetere, and C. Graham, "Designing TramMateña Context-Aware Mobile System Supporting Use of Public Transportation," in *Proceedings of the 2003 Conference on Designing for User Experiences*. San Francisco, California, USA: ACM Press, 2003, pp. 1-4.

[6] J. Dowling, R. Cunningham, A. Harrington, E. Curran, and V. Cahill, "Emergent Consensus in Decentralised Systems using Collaborative Reinforcement Learning," in *Post-Proceedings of SELF-STAR: International Workshop on Self-* Properties in Complex Information Systems*, *LNCS 3460*: Springer-Verlag, 2005, pp. 63-80.

[7] N. Honle, U. Kappeler, D. Nicklaus, T. Schwarz, and M. Grossmann, "Benefits of Integrating Meta Data into a Context Model," in *Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications Workshops*. Pisa, Italy, 2004, pp. 25-29.

[8] 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.

[9] M. Roman, C. Hess, R. Cerqueira, A. Ranganathan, R. Campbell, and K. Nahrstedt, "Gaia: A Middleware Infrastructure to Enable Active Spaces," *IEEE Pervasive Computing*, vol. 1, pp. 74-83, 2002.

[10] C. Borcea, C. Intanagonwiwat, P. Kang, U. Kramer, and L. Iftode, " Spatial Programming using Smart Messages: Design and Implementation," in *Proceedings of the Twenty-Fourth IEEE International Conference on Distributed Computing Systems (ICDCS'04)*. Tokyo, Japan, 2004, pp. 690-699.

[11] 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.

[12] 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.

[13] 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.

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

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

[16] 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.