2013

Exploring Spatial Business Data: a ROA Based eCampus Application

Thanh Thoa Pham Thi
Technological University Dublin, thoa-pham@tudublin.ie

Linh Truong- Hong
Technological University Dublin, linh.truonghong@tudublin.ie

Junjun Yin
Technological University Dublin, junjun.yin@tudublin.ie

See next page for additional authors

2013-04-01

Exploring Spatial Business Data: A ROA Based eCampus Application

James Carswell

Exploring Spatial Business Data: a ROA Based eCampus Application

Thanh Thoa Pham Thi, Linh Truong-Hong, Junjun Yin, James D. Carswell

Digital Media Centre, Dublin Institute of Technology, Ireland

{thoa.pham, linh.hongtruong, jcarswell}@dit.ie
yinjunjun@gmail.com

Abstract. In "Smart" environments development, providing users with search utilities for interacting efficiently with web and wireless devices is a key goal. At smaller scales, Google Maps and Google Earth with satellite and street views have helped users for querying general information at specific locations. However, at larger local scales, where detailed 3D geometries linked to business data are needed, there is a recognized lack of related information and functionality for in depth exploration of an area. Linking spatial data and business data helps to enrich the user experience by fulfilling more task specific user needs. This paper presents an eCampus Demonstrator for the National University of Ireland, Maynooth (NUIM) based on a Resource Oriented Architecture (ROA), in which various *RESTful* web-services have been developed and deployed for querying both spatial data and associated business data. The benefits and drawbacks of the chosen technologies are also discussed. This work can be considered as a platform that can be applied to similar application domains such as exploring business parks, hospitals, museums, etc.

Keywords: RESTful web-services, Resource Oriented Architecture, Mobile Spatial Interaction, eCampus Information Systems, 3D modeling

1 Introduction

Utilities for location-dependent queries are readily provided today by Google Maps and Google Earth with satellite and street views. However, at local scales, where detailed 3D geometries and associated business data are needed, there is a general lack of related information and search functionality for in depth exploration of an area. For instance, the following types of questions cannot be answered when interacting with Google Maps/Earth on a typical university campus: what classes are scheduled in that room over there? Whose office window is that up there? What specific buildings/classrooms/labs can I actually see around me while standing at a particular location on campus? Or in a hospital environment, by spatial querying hospital datasets we can ask what the surgery schedule is in that operating room? What medical equipment is installed in that room? What route should I take to move a bed or medical equipment from this room to another?

In order to answer these types of specific questions, Location Based Services (LBS) need to link spatial data with non-spatial business data. Spatial data deals with detailed topology and geometry or coordinates of objects while business data can describe the business semantic aspect of a related object in some business domain, e.g. a university campus.

In general, geospatial data does not change that much over time but business data seems to change faster depending on the nature of the business. Furthermore, conventional business data is often managed and produced by enterprise information systems independently of any geospatial applications. Thus linking spatial data and business data in one application helps to enrich the user experience by fulfilling more specific user needs. In particular for task specific decision making applications that need access to detailed local scale data typically found in Zoos, Museums, Hospitals, a University Campus or Business Park settings. Business data is often indirectly or virtually associated to spatial data via its location given by a generic address, a room number, a building name, or even lat/long coordinators of the business location. Such business data can provide further detailed information to users about the objects in question and be tailored or application domain dependent. For instance, in a university campus environment, the business data involved may be in the form of class schedules for a specific room, lists of equipment installed in a lab, office hours or contact details for a lecturer, today's special meal deal in a cafeteria, etc..

Typically, 2D footprint data provides just flat geometry representation of physical objects (e.g. buildings) in the horizontal plane. But for our application we need spatial and non-spatial attribute details in the vertical dimension as well for in-depth 3D BIM (Building Information Modeling) data query operations. Depending on how much 3D detail data is captured and how available any related business data is, more relevant task specific answers can be provided to the user. For instance, 3D BIM data of a building can include detailed digital representations of physical and functional characteristics for its different floors, rooms, windows, and doors where all aspects are potentially available for interrogation. In contrast to our approach of exploring such detailed 3D data, in [11] the authors also reconstruct 3D models of a campus but in this case it stays at the building level, where the room details have not been fully developed, and the implementation does not provide utilities to further question the area beyond its physical, spatial nature.

Within the framework of the StratAG project (http://stratag.ie/), we developed an eCampus Demonstrator for the National University of Ireland, Maynooth (NUIM). This web and wireless based GIS application aims to assist users in exploring and analyzing their surroundings within a detailed data environment; in our case domain specific business data is provided together with 3D spatial data to answer more specific user queries. The application addresses two types of users: public users (e.g. visitors) and local users (e.g. students and staff). Access privileges and query levels depend on user type. For example, visitors are presented with a campus map for general information querying of buildings and rooms, and utilities for general navigation to various buildings or departments. Meanwhile, in addition to these functionalities, staff and students are able to visualize their class schedules together

with personalized navigation plans and recommendations based on their academic and social interests.

Each partner in the StratAG cluster is responsible for developing different functionality in the Demonstrator such as utilities for directional querying the area, for path navigation assistance, for user interests recommendation, or for 3D modeling of the campus itself. *RESTful* web-services [7] were chosen for the deployment technology of these distributed components thanks to its simplicity when applied particularly to the geospatial domain [8].

In this paper, we present the architecture of our eCampus Demonstrator based on a Resource Oriented Architecture (ROA) model as it consumes and integrates the *RESTful* web services mentioned above. We focus on 2D and 3D web services for querying various distributed university datasets and for 3D modeling of campus buildings. The remainder of the paper is organized as follows: Section 2 gives and overview of the *RESTful* web-service concept and the design principles of ROA; Section 3 follows with a description of our eCampus Demonstrator application based on the ROA where different layers of the architecture are explained in more detail; Section 4 discusses some strengths and weaknesses of today's available technology for building such applications; and Section 5 draws conclusions and gives some direction for future work.

2 Resource Oriented Architecture (ROA)

ROA is "a paradigm for organizing and utilizing distributed *resources* that may be under the control of different ownership domains" [8]. Those distributed *resources* are called *RESTful* web-services. *REST* stands for the *Representational State Transfer* style that is "an abstraction of the architectural elements within a distributed hypermedia system" [4, 7]. It relies on basic web protocols HTTP and MIME (Multipurpose Internet Mail Extensions) [9]. Some main concepts of *REST* can be found in [7] as follows: *Resource,* which is the central concept, is "the intended conceptual target of a hypertext reference". Anything can be a resource such as a document, an image, physical object, or a collection of other resources, etc.; The *resource identifier* is the hypertext link Uniform Resource Locator (URL) or Uniform Resource Identifier (URI). When the link is selected, the resource will be moved from where it is stored to where it is used by HTTP.; The *representation* in *REST* is a sequence of bytes, plus *representation metadata* to describe those bytes such as a text file or an image; In addition, the *REST* implementation requests some constraints such as *Uniform Interface*, *Self-Describing Messages*, *Hypermedia Driving Application State* and *Stateless Interactions* which are discussed in detail in [1].

A *RESTful* web-service is any web-service that implements the *REST* architecture style and is identified by a URI or URL link on the web. Calling a *RESTful* web-service is done simply by accessing its resource identifier through the HTTP protocol. There are typically five HTTP operations available to *RESTful* web-services [4]: PUT (create a new resource), GET (retrieve the representation of the resource), DELETE (delete the resource), POST (modify the resource) and HEAD (obtain meta

information about the resource). For instance, when calling a *RESTful* web-service with the GET operation, the service returns (gets) the results (resource representation) in popular formats such as XML, JSON (JavaScript Object Notation), or KML (Keyhole Markup Language) format. The following example illustrates calling a *RESTful* web-service with Ajax-Javascript and retrieving the resource in JSON format:

```
$.ajax({
  type: "GET",
  url: URIstring, // URI of the RESTful web-service
    with its parameters
  dataType: "json",
  success: function(js){
    // process js
  },
  error: function(e){
    alert(' Error ' + e.status + ' : ' +e.statusText);
  }
});
```

In the scope of our demonstrator, the system aims at providing answers to user queries that demand retrieval of both spatial data and business data. Therefore only the GET operation is used.

3 ROA based eCampus Application

Our NUIM eCampus Demonstrator is a web-based application accessible to both desktop and mobile devices. It aims to help users explore in more detail the university campus by providing them maps and utilities for both 2D and 3D querying. Different query functionality is provided so that users can ask questions by interacting with the map itself. For instance, they can ask: What is that building over there by pointing at it with their mobile device; What is the class schedule of this classroom by clicking on it; What can I actually see around me when standing in a particular location on campus, etc. The query results are displayed on the map afterwards together with links to further business data where available. This process is repeated and illustrated in Figure 1 in UML (Unified Modelling Language) activity diagram notation as follows:

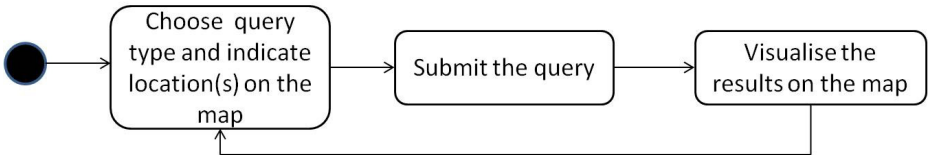

Fig. 1. The workflow of user interactions

The application invokes various *RESTful* web-services to access the various query functionality. A catalogue of the services is provided in the form of a list of URI services with a description of their functionality and required parameters. The application architecture is depicted in Figure 2. It includes 3 main layers: Database Layer; Web-Services Layer; and the Interface Layer.

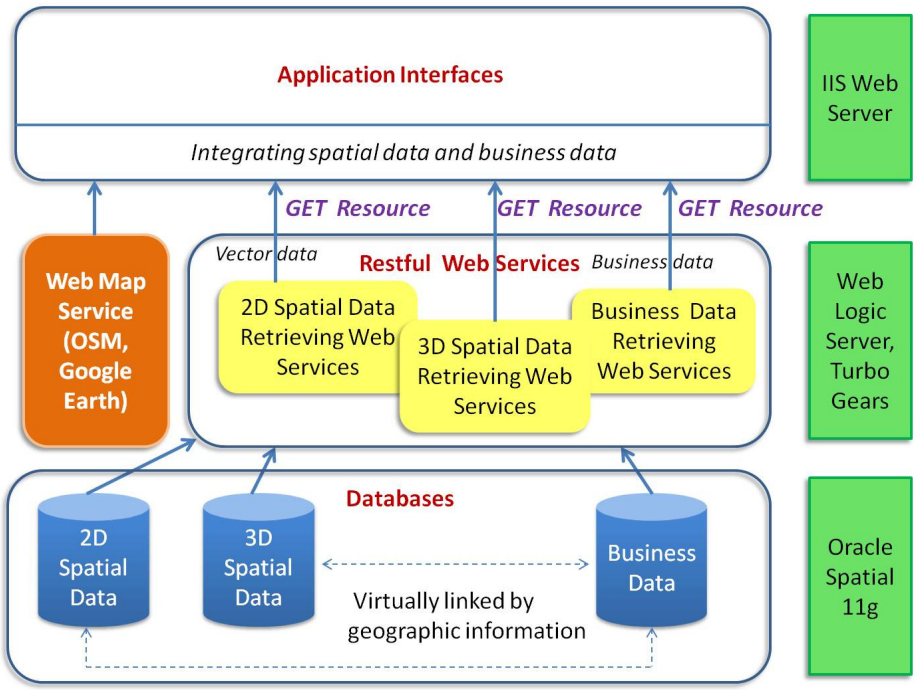

Fig. 2. eCampus Demonstrator architecture based on ROA

3.1 Database Layer

The 2D footprint and detailed 3D models of NUIM campus are hosted in Oracle Spatial 11g databases that also serve the spatial data retrieving web-services. For visualizing 3D buildings in Google Earth for querying, corresponding KML files are prepared. This preparation process will be discussed in more detail at the end of this section.

2D database preparation.

Many online maps such as Google Maps and Bing Maps have considerably poor data coverage over many areas especially in those less populated, like Maynooth Ireland. As NUIM's campus is located in such a rural area, OpenStreetMap (OSM) shows the advantage of collective efforts for Volunteered Geographic Information

(VGI), where anonymous users contribute to OSM by uploading geographic features such as buildings, streets, point-of-interests (PoIs), etc. to complete the map coverage. In particular, students and others from NUIM have already done this, which proved very beneficial in our case [17]. Therefore, many 2D campus footprints were first downloaded from OSM and then uploaded to our local Oracle Spatial DBMS where geometry data is stored in a single column data type of SDO_GEOMETRY which defines the geometry type (e.g. points, lines, polygons, solids, etc.), the dimension, and an array of x, y (and z for 3D) coordinates comprising points or vertices of campus objects.

3D database preparation.

According to [16], volumetric objects like 3D buildings can be represented by polyhedrons, triangulated polyhedrons or tetrahedrons of which the polyhedron is the most appropriate representation for buildings. As such, the polyhedron that bounds a solid comprised of multi-polygons [17] used to construct a 3D building model is adopted in our work. Meanwhile, a simple polygon is used to represent the 2D footprint of a room. Figure 3 describes the process of capturing 3D building data and transferring the data into the Oracle Spatial databases used in our application.

Firstly, the 3D building models were manually reconstructed from terrestrial laser scanning data by using proprietary programs of the scanner and CAD environments where point cloud data gets geo-referenced before post-processing. 3D solid objects represent the various building components and polygons are created from the borders of windows and doors in order to support spatial queries of objects at a detailed level. Further room attribution derived from architectural drawings is also assigned to the extracted room level data. In summary, 3D solids and 3D polygons respectively represent the campus building models and their associated rooms linked to any available metadata information. The developed database schema consists of two tables named NUIMBUILDING and NUIMBUILDINGOBJECT, where the first table contains the geometry of the buildings, while the second stores the geometry of the associated windows/doors (Figure 3).

Additionally, for exporting 3D objects in an AutoCAD environment, the Feature Manipulation Engine (FME) workbench utility was used. A workflow for exporting data from a DWG file of AutoCAD into Oracle Spatial was developed and is shown in Figure 4. The upper right window depicts the workflow to transfer the external objects (exterior walls, roofs, balconies and window frames) of the buildings into the table NUIMBUILDING. These objects were stored in a row of that table, which are underlying multisurface (SDO_GTYPE = 3007). The lower right window illustrates the workflow to transfer the polygons on peripheries of all windows of each room to a row in the NUIMBUILDINGOBJECT table as heterogeneous (SDO_GTYPE = 3004).

Note that at this phase of the project, objects inside buildings are not modelled and captured. A room is represented by its windows, selecting a room corresponds to selecting one of its exterior windows (or doors). Therefore our spatial database contains mainly building objects and associated window/door objects.

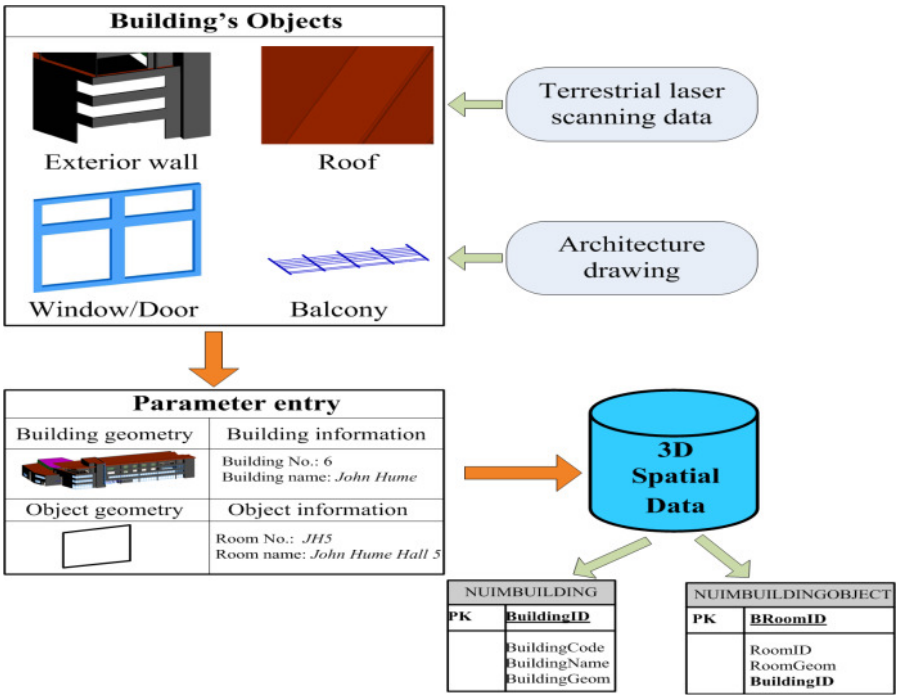

Fig. 3. Process of 3D data development in Oracle Spatial 11g

KML file preparation.

The process of converting data utilizing FME workbench is shown in Figure 5, where 3D geometries are made readable by Google Earth by first transforming them to a KML/KMZ file as 3D polygons through two schemes. The first scheme (upper middle window in Fig 5) deals with the creation of all external objects of a building (e.g. exterior walls, roof, doors, balcony and window frames) associated with general information (e.g. building name) through the AttributionCreator Function of FME workbench. In the second scheme (lower middle window), polygons of window/door peripheries and associated "room" attribution were loaded into a KML file, in which the room name was directly extracted from CAD drawings. In this work, information about the building and its related rooms were defined through a transformer KMLPropertySetter while rendering was carried out by manually assigning colors for each geometry object via a transformer KMLStyle in the FME workbench utility. Loading general information about buildings and its rooms (windows/doors) into a KML/KMZ file is used for display when users click on those objects in Google Earth.

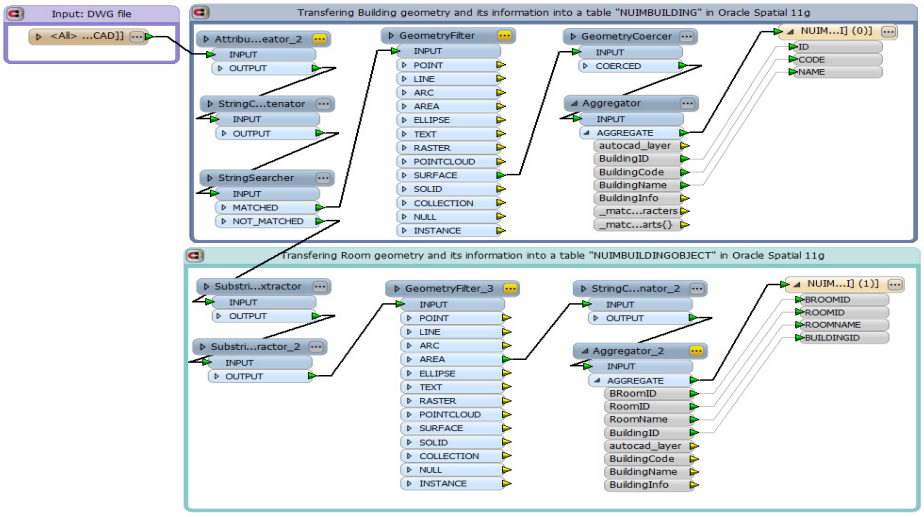

Fig. 4. Process of transferring geometries of buildings and associated windows/doors together with their metadata information into Oracle Spatial 11g.

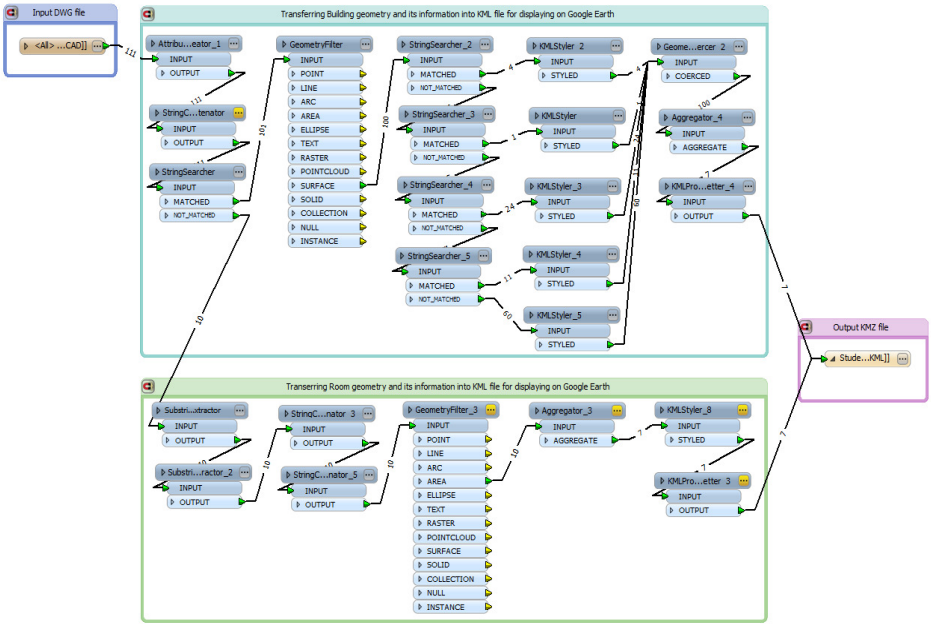

Fig. 5. Process of transferring geometry of a building and associated windows/doors into a KML file suitable for upload to Google Earth.

3.2 Web-Services Layer

This layer relates to the development of spatial data retrieving web-services and business data retrieving web-services.

Spatial data retrieval web-services.

The collection of spatial functions for performing 2D and 3D visibility based directional querying are deployed in the form of *RESTful* web-services. In relation to the different spatial data formats found in the database, the web-services are divided into two sub-groups, one applied to 2D campus footprints and the other applied to 3D campus models. A diagram illustrating detailed components of the various web-services is shown in Figure 6. All neighbors, Field-of-view, Line-of-Sight and Threat dome are just a few of the different types of spatial queries available to the user where different search algorithms and web-services were developed and deployed for each query type [13, 14].

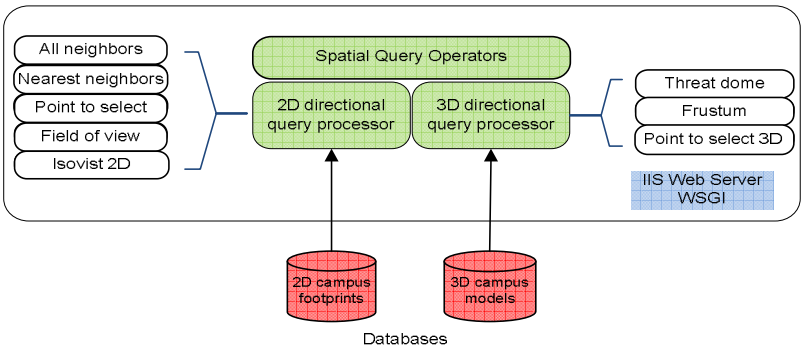

Fig. 6. System architecture for delivering 2D/3D spatial query services

More specifically, an IIS (Internet Information Services) server is appointed to host the web-services. Query requests are constructed using standard HTTP calls containing a valid URL filled with the required query parameters, like the location of a mobile device. A Web Server Gateway Interface (WSGI) was used for Python programming, which defines a universal interface for communications between the web server and web applications. Below are two example URLs constructed for requesting a particular web-service:

URL1:
http://ecampus.dyndns.org/fov/?tg_format=JSON&lat=53.3387&lng=6.2675&heading=44.0&angle=30.0

URL2:
http://ecampus.dyndns.org/point2select3d/?tg_format=JSON&lat=53.383522033691406&lng=-6.60053253173828&heading=44.0&tilt=30.0

URL1 is for the "field-of-view" query request, where (fov) in the string is the function name, the specified response format is (JSON) whilst the required parameters are: the user's coordinates in the form of latitude (lat) and longitude (lng), their heading or pointing direction (heading) and the size of the view angle (angle).

URL2 is for the "point-to-select" or "line-of-sight" query request in a 3D environment. The format for a 3D query URL is almost the same as in 2D. However, because it involves the vertical dimension, the last parameter in this function is the tilting angle (tilt) of the mobile device (taken from the accelerometer) at a particular location (taken from the GPS receiver) while maintaining the heading direction (taken from the compass). For the desktop version of this 3D query, the above parameters are input by the user directly on the Google Earth interface.

The response from the service is wrapped in JSON format which is essentially a text based key-value set. For example, the result returned from a "field-of-view" 2D web-service call is presented below. This returns all objects intersecting the user's viewshed polygon at the current location, view angle, and heading with its attribute information listed as name, latitude and longitude respectively. It is up to the application developer to further parse this information and visualize accordingly depending on end user requirements.


```
{"Result":{"items":
[["name" : "John Hume Building",
"coordinate" : { "lat": 53.3833658533,"lng":
-6.5994806592},"shape": [-6.6004514, 53.3841367,
-6.60037709999995, 53.3840353000001,...,
-6.600309499999979, 53.3842268000001, -6.6004514,
53.3841367]], ["name" : "Saint Catherines Building",
"coordinate" : { "lat": 53.383704272, "lng":
- 6.5991442043}, "shape": [-6.5966732, 53.3821449, -
6.5966932, 53.3816629,..., -6.5961998, 53.3821441, -
6.5966732, 53.3821449]]]}}
```


Business data retrieval web-services.

The business data retrieval web-service is used for querying schedules of class rooms, retrieving equipment lists of labs, or for retrieving timetables of students from separate databases. In our application, such business data also happens to be stored in Oracle 11g but maintained by NUIM independently. The web-services for retrieving business data are also *RESTful*, and are hosted on an Oracle Web Logic server. For example, the following web-service call returns the current class schedule for room 1_02JH (1st floor, room number 2, in John Hume Building) in JSON format indicating date, time, and class modules taking place in that room for the week:

URL:

http://147.252.87.151:7101/getSchedulecontextroot/jersey/Schedule/getS?roomID =1_02JH

Returns:

```
{"schedule" : {"items":[["Monday",9,"MN304"],["Monday",
11, "MT211S"], ["Tuesday",10.30, "HY210"], ["Thursday",
13, "EC201"]]}}
```

In our approach, the integration of web services is performed at the interface level. This is presented in more detail in the following section.

3.3 Interface Layer

The integration of spatial data and business data is performed at the client side when information about buildings and associated rooms is found and returned. The retrieved spatial data is visualized on a map as additional layers for 2D queries (using OpenLayers API). Meanwhile for 3D queries, the results are returned in JSON format and drawn as Placemarks added to a Google Earth view.

Once the room(s) detail, including room number, is extracted from the result set, it is filled into the URI for calling the getSchedule web-service, then this response is added to a *Balloon feature* of the Placemark so that the corresponding business data gets displayed whenever the room is highlighted as showed in the following code:

```
var balloon = ge.createHtmlStringBalloon('');
balloon.setFeature(polygonPlacemark);
  // polygonPlacemark is the window viewed
balloon.setContentString(content);
  // content is business data
ge.setBalloon(balloon);
```

Figure 7 illustrates a directional query from the user's current position and pointing to a specific room in a building using the Point-to-Select web-service. The yellow line illustrates the user's 3D pointing direction, the coloured window represents the room. When the users click on that window, the class schedule of the corresponding room is displayed. In case of a Frustum query, it depends on the provided horizontal angle and vertical angle so that any query response may involve many rooms.

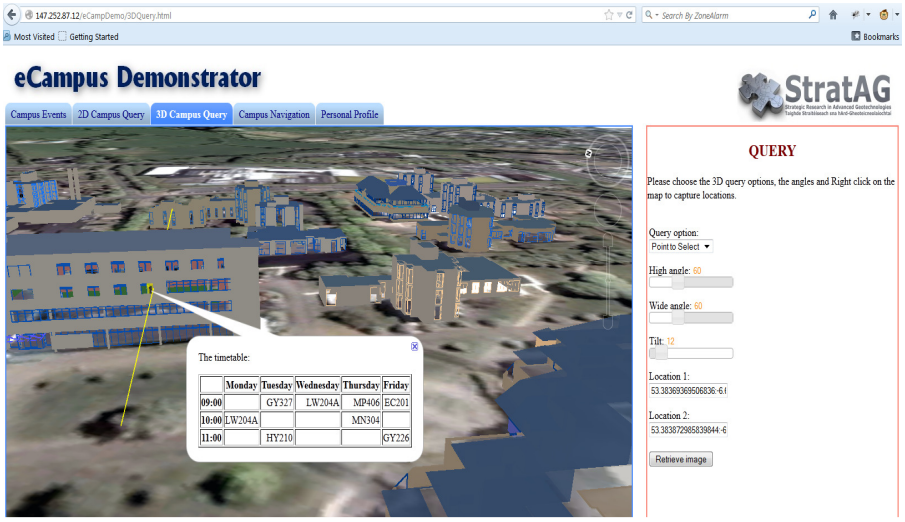

Fig. 7. The result of a Point-to-Select 3D query showing linked business data (class schedules) retrieved for that room in a *Balloon feature* within Google Earth

Figure 8 shows the result of a 360˚ Isovist query in 2D. Users can either click on the map while interacting with the desktop version of the application or use their mobile GPS reading when interacting with the wireless version to indicate a location, the red polygon represents the viewshed of the user in 360 degrees out to a user defined radius. The blue polygons represent the objects that intersect the Isovist (i.e. what users can actually see from their current location in all directions).

Fig. 8. The result of a 360˚ Isovist 2D query

4 Discussion

During the development of this application, some important limiting conditions emerged regarding the technology employed. After analyzing the capabilities of the chosen technologies for implementation, the following discussion presents some advantages and disadvantages of our approach.

Integration level of spatial data and business data.

As mentioned previously, the integration of spatial data and business data is performed at the client side, i.e. at the visualisation level. We considered three options to provide spatial data and related business data to users as shown in Figure 9.

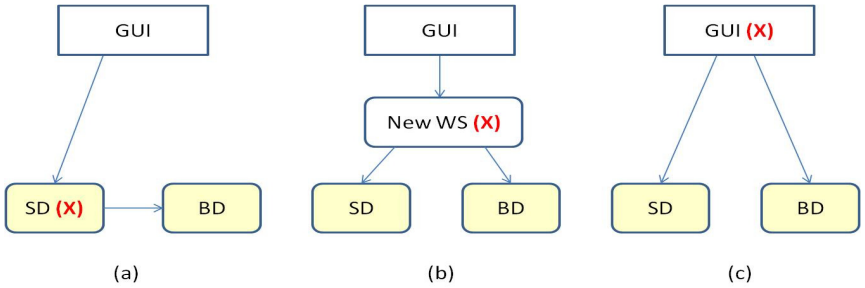

Fig. 9. Different spatial data and business data integration approaches. The red (X) represents the integration point. The arrow describes the calling direction. WS: Web-Service; SD: spatial data WS; BD: business data WS; GUI: Graphical User Interface

In Figure 9 above:

(a) Early-integration: In this approach, retrieving business data is performed from inside the spatial data retrieving WS. The final results sent back to GUI include spatial data and business data.

(b) Aggregated web service: A new web-service is developed to compose the results returned by the spatial data web-service and the business data web-service.

(c) Integration at the visualisation level: The results of the spatial data web-service and business data web-service are overlaid at the visualisation level. We have chosen the third approach: integration at the visualisation level, as this approach provides some advantages compared to the other two approaches (Table 1).

OGC services.

The Open Geospatial Consortium (OGC) has developed some standards for geospatial processing technologies to enable applications from different commercial vendors to interoperate. However, the locationing services developed within OGC focus mainly on tracking and location-based applications for mobile devices [10]. However, these services are far from what we require in this application, where all

RESTful web-services for location-dependent directional and Vista space querying have been developed in our own algorithms.

Table 1. Spatial data and business data integration options

(a) Early-integration	There is a dependency of SD retrieving WS and a specific BD retrieving WS. That means SD WS cannot be reused for other purposes
(b) Aggregated web-service	SD and BD WS are independent. It depends on the user needs that the aggregated WS (AggWS) will integrate suitably SD and BD WS. Suppose *fA(WS)* is the cost of analyzing the result of a WS, *fC(WS)* is the cost of calling a web-service, then the cost of this approach to display the result is: *fC(AggWS)+ fA(Agg WS) + fC(SD)+ fA(SD)+ fC(BD)* Note that the results of AggWS needs to be analyzed to draw the geometry shapes and the business data is then added to the feature data of the geometry object. In case the results of AggWS is in KML format, there is no need of analysing AggWS (so no cost), all spatial data and business data can be visualized. However, in that case the visualisation is fixed according to the API provided.
(c) Integration at visualisation level	SD and BD WS are independent. It depends on user needs that suitable SD and BD WS are consumed at the visualisation (client side code source). The cost to display the result to the users is: *fC(SD)+ fA(SD)+ fC(BD)* Visualisation of the results is flexible according to the users' needs.

ROA instead of SOA.

For the last decade, Service Oriented Architectures (SOA) have been widely used for distributed applications, particularly on the Web. In Geographic Information Systems (GIS), there is no exception here. For instance in [2], a GIS web-service architecture was proposed based on SOA technology. However, while SOA is a proven approach, in some cases it can be overly complicated thus processor heavy. For example, when handling a SOAP message, the client (desktop or mobile) needs to send a request with parameters constructed and wrapped in XML format with special headers and other elements. It also has to parse any response from the server in the same effusive XML format [6]. But in the case where the client is a mobile device, this approach contains far too much processing overhead in terms of the volume of data, most of it quite unnecessary, that must be sent/received on mobile devices having relatively limited wireless connection speeds and often a data transmission cost [12]. In this respect, JSON is a much lighter data format in terms of processing and transmitting wirelessly. Furthermore, we agree with the general statement that the *REST* architecture provides a "scalable and simple deployment of web services and

particularly appealing for Earth and Space Science" [8] as *RESTful* web services have been much used in geo-information sharing.

Dependency of 3D query performance and 3D data details.

In our application, users carry out spatial queries from outside buildings. Therefore only the geometries of exterior structural components of the building (e.g. facades, roofs, windows, doors, balcony, canopy and so on) associated with room level attribution (e.g. room name and function) were loaded into the database. In this way, it helps to reduce the complexity of the 3D models and thus improve 3D spatial query performance.

Limitations of the length of URI link.

In theory, the HTTP protocol does not limit the length of a URI (according to RFC 2616 - Hypertext Transfer Protocol - HTTP/1.1). However some browsers may limit extremely long URI text strings, therefore the application should take into account this issue when designing *RESTful* web-services.

5 Conclusions and Future Work

Providing users with business context data in location dependent queries helps to fulfil more users needs within detailed data environments. The flexibility, interoperability and heterogeneity of this kind of linked search application demands a suitable software architecture. In particular to this geospatial application, a Resource Oriented Architecture (ROA) was chosen for the implementation.

The eCampus application allows users to explore the NUIM campus in 2D and 3D by interacting with Open Street Map and Google Earth interfaces. We also discussed the benefits and drawbacks of the chosen technologies. This work can be considered as a platform for developers and researchers when developing for similar application domains, such as when exploring a business park, hospital, or a shopping centre, etc.

In the next version of the Demonstrator, we address more advanced query types to support more specific user needs when exploring the campus in 3D mode. For example, the system could answer queries about a specific building such as, "Show me all class rooms that are available (i.e. not booked) from 9-10am?". These kinds of queries need more business web-services to retrieve and analyze the required business data. Consequently, the user interfaces also need to be improved to allow for asking these more detailed questions.

In the near future we aim to test the application in the hands of real students/staff/visitors to NUIM Campus on mobile devices which, apart from interacting with the map to indicate specific locations and ad hoc directional queries as usual, real-time user location and personal preferences can also be gathered and exploited as input query parameters to other StratAG partner web-services included in the complete eCampus Demonstrator application. Enhancements to graphical object interaction functionality within the desktop Google Earth interface will also be investigated.

Acknowledgements

Research presented in this paper was funded by a Strategic Research Cluster Grant (07/SRC/I1168) by Science Foundation Ireland under the National Development Plan. The authors gratefully acknowledge this support.

References

1. Guinard, D., Trifa, V., Wilde, E.: A Resource Oriented Architecture for the Web of things, IEEE International Conference on Internet of Things, Tokyo, Dec (2010)
2. Alameh, N.: Chaining Geographic Information Web Services, IEEE Internet Computing, September October (2003)
3. Lucchi, R., Millot, M., Elfers, C.: Resource Oriented Architecture and REST, JRC Scientific and Technical Report (2008)
4. Fielding, R. T., Taylor, R. N.: Principal Design of Modern Web architecture, ACM Transactions on Internet Technology, Vol. 2, No. 2, pp.115-150 (2002)
5. OpenGIS Abstract Specification Topic 12: OpenGIS Service Architecture (2002)
6. Snell, J., Tidwell, D., Kulchenko, P.: Programming Web Services with SOAP, O'Reilly Publisher, December (2001)
7. Fielding, R. T.: Architectural Styles and the Design of Network-based Software Architectures, PhD dissertation, University of California, Irvine (2000)
8. Mazzetti, P., Nativi, S., Caron, J.: RESTful Implementation of geospatial services for Earth and Space Science applications, International Journal of Digital Earth, Vol. 2, Supplement 1, pp.40-61 (2009)
9. Kurtagic, H., Birch, J., Zeiss, G.: An Open Architecture for RESTful Geospatial Web Services, FOSS4G Sydney (2009)
10. *OGC* Report- Summary of OGC Web Services, Phase 4, Interoperability Testbed (2007)
11. Kaharaman, I., Karas, I. R., Rahman, A. A.: Developing web-based 3D Campus Information System, ISG & ISPRS (2011)
12. Yin, J. and Carswell, J.D.: Touch2Query enabled mobile devices: a case study using OpenStreetMap and iPhone. In Web & Wireless Geographical Information System (W2GIS 2011) Springer, Kyoto, Japan, pp. 203-218 (2011)
13. Carswell, J.D. 3DQ: Threat Dome Visibility Querying on Mobile Devices. GIM International, Vol.24, (8), 24, August 2010
14. Carswell, J., Gardiner, K., Yin, J.: Mobile Visibility Querying for LBS. Transactions in GIS Journal, 14(6): 791–809. 2010.
15. Jacob, R., Zheng, J., Ciepłuch, B., Mooney, P. and Winstanley, A.C.: Campus Guidance System for International Conferences Based on OpenStreetMap. In Proceeding W2GIS '09, Maynooth, Ireland, Springer-Verlag, pp. 187-198 (2009)
16. Zlatanova, S.: 3D geometries in spatial DBMS. 3D-GIS, pp.1-14 (2006)
17. Musliman, I. A., Abdul-Rahman, A., Coors, V.: Incorporating 3D-GIS Spatial Operator with Building Information Models in Construction Management using Geo-DBMS. 5th International Conference on 3D GeoInformation, 3-4 November 2010, Berlin, Germany, Volume XXXVIII-4/W1, 147-154 (2010).