The ITB Journal

Volume 5 | Issue 1 Article 13

2004

Architecture and development methodology for Location Based Services

Aaron Hand
School of Science, Institute of Technology at Tallaght, Dublin 24., aaron.hand@itnet.ie

Dr. John Cardiff
School of Science, Institute of Technology at Tallaght, Dublin 24.

Architecture and development methodology for Location Based Services

Aaron Hand[1], Dr. John Cardiff[2]
[1] *School of Science, Institute of Technology at Tallaght, Dublin 24*
[2] *School of Science, Institute of Technology at Tallaght, Dublin 24*
Contact email: aaron.hand@itnet.ie

Abstract

This paper presents a LBS (Location Based Service) architecture, development methodology and a development tool to assist LBS developers in building new open standard LBS applications. The approach adopted has been to define new LBS systems based on open standards rather then proprietary driven technologies. SAGESS (Spatial Application Generic Environment System Standards) is an architecture platform for the development and deployment of wireless LBS applications. SAGE (Spatial Application Generic Environment) is a development methodology that outlines a step-by-step development approach to LBS application development. A prototype LBS application was deployed on the author's SAGESS architecture platform, developed using the author's SAGE development guidelines and the SAGE development toolkit. SAGESS and SAGE will decrease LBS development difficulties and provide an open standard approach to LBS application development.

Keywords: Mobile computing, LBS (Location Base Services) applications, LBS Architecture, LBS methodology.

Introduction

The emergence of location technology as potential new market revenue, the next "killer applications" (Albena Mihovska & Jorge M.Pereira, 2001) has caused existing GIS (Geographic Information Systems) application software and telecommunication technologies to try and combine this new technology in an unified mix and match style of architectures. Therefore an open standard architecture framework is required for future market growth and stability of the LBS/GIS market place. The SAGESS LBS architecture and SAGE development methodology goals are to solve existing wireless LBS problems and establish an open frame base for the development and deployment of wireless LBS applications. The research described in this paper outlines an LBS deployment architecture and LBS development methodology, to decrease the difficulty in LBS application development and deployment. This project also defines a LBS development toolkit in order to assist first time and existing LBS developers to develop flexible LBS applications for the present (2004) wireless devices and future wireless devices. The paper is organised as follows: SAGESS and SAGE overview followed by a prototype demonstration.

SAGESS Architecture

The SAGESS platform is a three-tier architecture. The first tier is the client communication tier, second tier is the application tier and the third tier is the GIS information tier. The SAGESS architecture incorporates findings of Rui José, Filipe Meneses & Geoff Coulson, (2001) and Bharghavan, V. & GUPTA, V., (1997) and Jin Jing, Abdelsalam Sumi Helal & Ahmed Elmagarmid, (1999).

Client Communication tier

The client communication tier is a protocol independent tier where the users location is established and communication with the application tier occurs. The user launches an Internet browser and makes a wireless Internet connection to the application tier. The user invokes the LBS application by entering the application URL (Uniform Resource Locator). Any wireless location method that returns the geographical location of the user can be used to locate the user's current location. The wireless communication between the client communication tier and the application tier can be any form of wireless Internet protocol communication or technology.

Application tier

The application tier performs all result-set mark-up (voice, text, image). The result-set mark-up is the output displayed from the LBS solution. The application tier has two internal components, an *XSLT* (eXtensible Stylesheet Transformations) *processor* and an *SQL/XML translator*. The application tier receives over a wireless Internet connection the user's current location, if the user's current location cannot be obtained at the client communication tier then it can be obtain at the application tier by means of either third party software or a MLC (Mobile Location Center). The location can also be determined at the GIS tier with location sharing between the telecommunications database and the GIS database. The application tier *SQL/XML translator* component communicates to the GIS tier the current user's location and the LBS application query. The *SQL/XML translator* transforms the result from the LBS application query into an XML formatted document. The *XSLT processor* transforms the XML document into a specific or a range of different mark-up formats (SVG, HTML,PDF,WML,VRML). The resulting outcome is then displayed in the user's Internet browser.

The application tier in the SAGESS architecture will transform dynamically generated XML into any desired mark-up output. The XSL applied to the XML document decides the output of the data in the XML data page. One XML document can have many different XSL depending on screen size, browser language (English, French) and preferred mark-up Language

content (HTML, WML, SVG, VRML). The application server can automatically discover the browser type of a calling application and automatically assign the appropriate XSL.

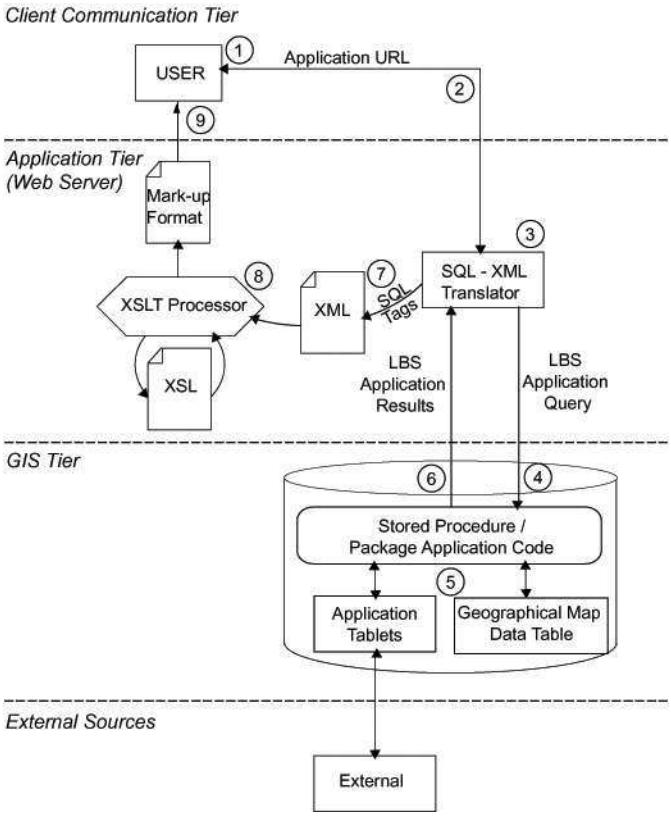
GIS (Geographic Information System) tier

The GIS information tier will perform all application query processing. The GIS tier stores, modifies and retrieves spatial data sets based on the LBS application query from the application tier. The application tier evokes an *application specific database stored procedure or package* which executes the core application logic. The *application specific database stored procedure or package* interacts with *geographical map data tables* (digital maps in relational formats) to perform LBS spatial related queries. The *application specific database stored procedure or package* interacts with the *geographical map data tables* and with the *application tables*. Application tables contain data that is utilised by LBS applications to perform more user specific and accurate LBS applications. *Application tables* can be a range of tables each table obtaining information from many different outside sources. *Application tables* could contain data on traffic information, road works, amenities location (ATM's, shopping centres, car parks) and services (restaurants, bars, cafés, hotels). These tables can be automatically and dynamically updated from existing external sources. The tight coupling of core application code and dynamic *application tables* allows for the creation of highly adaptive LBS applications. The GIS tier requires a database that can manipulate spatial data.

below displays on overview of the SAGESS architecture and demonstrates the message flow through the system. The steps of communication are:

1. User enters the HTTP address of the application. The user's location can be determined at this point or at a later point. **(Client Communication tier)**
2. The Web server accepts the user request and offers LBS application options if present. **(Application Tier)**
3. The *SQL/XML translator* makes a JDBC/ODBC connection to the database on the GIS tier with the LBS application query and user location. **(Application Tier)**
4. The core application logic (stored procedure/package) is evoked by the LBS application query sent from the *SQL/XML translator.* **(GIS Tier)**
5. The stored procedure/package utilises many application specific tables and manipulates spatial data to produce the application result set. **(GIS Tier)**
6. The application SQL result set is returned to the calling *SQL/XML translator* component. **(GIS Tier)**
7. The *SQL/XML translator* converts the SQL result set into an XML document with appropriate name tags. **(Application tier)**
8. The XSLT processor then transforms the XML document into the mark-up page specific by the XSL. **(Application Tier)**
9. The user's browser can then interpret the output from the mark-up page. **(Client Communication tier)**

below the numbers represent the steps above.



SAGESS Architecture

The SAGESS three-tier architecture design provides clear coupling of resources and a more flexible and scalable architecture then a two-tier approach. This style of architecture allows for development changes to occur on the server side and for thin accessing clients. Similar style of architectures are used in many commercial LBS systems and in the PARAMOUNT (Poslad S, Laamanen H., Malaka R., Nick A., Buckle P. & Zipf, A, 2001) and CRUMPET (Loehnert E., Wittmann E., Pielmeier J., & Sayda F., 2001) research system. The architecture is based on the use of standard protocols (HTTP, TCP) and open standard database connection interfaces (JDBC,ODBC) to provide a more flexible and adaptive platform for future wireless LBS development. SAGESS is made up of software components that can be swapped and updated but the overall message parameters remain the same. The SAGESS architecture combines two different areas of computing GIS and LBS underneath a coherent flexible architecture.

SAGE Development Methodology

The SAGE development methodology is intended to assist LBS developers to prosper from the SAGESS architecture and deliver simple to develop, flexible, device independent attractive LBS applications. The SAGE development methodology deals with issues of user data modelling (interface layout and language representation), complex functionality, geo-data processing, generic application development and LBS development environment design.

SAGE deals with one of the key LBS development problems of third party technology and not methodology being the driving force behind LBS application development. SAGE deals with LBS development of wireless database structure issues (Dunham M. H. & Helal A, 1995) and nomadic computing difficulties (Alonso R. & Korth H. F, 1998). The SAGE development methodology's principal factor is a database data centric approach to LBS application development. This data centric approach is necessary, as LBS applications are data intensive applications, which are accessed from limited wireless devices. The database data centric approach provides the developer with more flexibility, security and a stably controlled development environment.

SAGE defines clear development steps for LBS application development:

1. Database geographical tables.
2. Define application specific tables.
3. Develop core application functionality.
4. Define eXtensible StyleSheet (XSL) for result display.
5. Develop Internet based (GUI) Graphical User Interface to evoke LBS application

The SAGE development methodology couples the mandatory spatial map query with the core application logic. This enables all application function processing to be performed inside the database. The calling LBS application will receive the entire data result for the LBS application query. This style of LBS application function processing removes the application process cost, from the limited wireless devices to the large process GIS server, coupled with the large processing application tier.

The third step of SAGE is to develop core application functionality. The core LBS application functionality (application specific functions and location query function) are developed using database store procedures. Database procedures are implemented in a form of SQL (Structured Query Language). SQL is a standard interactive and programming language for manipulating data stored in a database. SQL is both an ANSI and an ISO standard. Database procedures define for application development should have:

- Parameters names related to their function.
- Common LBS functions implemented in dynamic SQL.
- A store procedure comments table set up.
- Transport parameter called transport.

A store procedure comments table should be set up inside the database that stores a LBS developer description of the store procedure. The description table will assist new developers in building LBS applications based on integrating existing procedure capabilities.
Dynamic SQL enables a procedure to assign dynamic values at run time. This feature enables stored procedure to adjust to varying databases conditions and one stored procedure can execute differently based on parameters. Dynamic SQL is ideally suited for repetitive task with changing parameters. Database procedures can now perform similar tasks to external programming languages. The advantages of database procedures languages over external languages are processing time and complex data type matches; two major factors in LBS applications.

Database store procedures can increase LBS processing speeds as store procedures are stored internally to a database in compiled code and procedure execution plans can become cached in memory vastly increasing execution time. LBS applications relay on a mandatory spatial data query. Database procedures perform this task quicker and with greater flexibility then any external procedure languages. Database stored procedure will execute identically in the same proprietary database, running on different operating systems (OS).

SAGE Toolkit

The SAGE toolkit is a wizard application to develop quick LBS applications based on the SAGE development methodology and deployed on the SAGESS architecture. The SAGE toolkit incorporates database spatial functions options and allows new LBS application to be developed based on existing development templates. The SAGE wizard toolkit will:

- Provide a wizard style GUI (Graphical User Interface).
- Incorporate XSL templates.
- Enable LBS query testing.
- Display database LBS query functions.
- List Procedure templates.
- Generate an automatic LBS application interface.

The SAGE development wizard will be initial set up with the default spatial settings for the proprietary database connection. LBS users can change these settings and all connection settings in the program. The SAGE development toolkit requires LBS developers to enter in the application specific LBS queries and the name of the appropriate XSL. The XSL used can be one of the existing templates or a new template based on the existing template.

SRF (Service Route Finder) Prototype Application

The SRF application is a prototype application that was developed using the author's toolkit on the SAGESS platform using the SAGE development methodology. The SRF application combines both LBS/GIS into a dynamic mobile solution for locating services (Restaurants, ATM's, Hotels, Cinema, etc.) on low resource devices over standard communication protocols. This chapter will detail the implementation structure and features of the SRF application.

SRF Application

The SRF is an LBS prototype application that was implemented on industrial technologies based on the SAGESS architecture. The SRF application performs a wide range of user and location based services.

The SRF application:

- Finds services that are geographically local to the user.
- Finds the shortest route to the service location.
- Calculates the route based on method of transport and on up to the minute traffic information, and road maintenance information.
- Open standard content, non-application specific.
- Dynamic user specific generated map images.
- Estimates arrival time, and distance along the route path.
- Provides a rerouting service back to the users original location.
- Provides detailed information on the selected service.
- Provides additional LBS services like nearest ATM, car park, monument on route.
- Incorporate external information systems to improve solution accuracy.

The SRF application delivers user specific maps in SVG format. The SVG maps are automatically generated and can be displayed on any Internet browser device that has SVG support. SVG can be dynamically generated by any of the proposed methods of John McKeown, & Jane Grimson (2000). SVG is a vector-based image therefore it can be scaled and altered without losing image quality. This feature will resolve the wireless development issues of delivering a solution to multiple wireless devices with different screen sizes. The SRF application generates maps with information that is specific to the user's request. The generated maps only show information (roads, ATMS, Car Parks, Monuments) situated on the users route (Dublin City Walking Nearest Restaurant). This feature allows for more tailored made service information and SVG images of lower size. The generated maps contain Internet-based intelligence of restaurant Internet site hyperlinks, dynamic travel time, voice description and name enlargement. The generated user adaptive maps will improve tourism and service location as just the information required is delivered in a clear and concise manner as described by Alexander Zipf, 2002. The SRF application can also deliver hypertext based information for devices that not SVG capable.

SRF Architecture

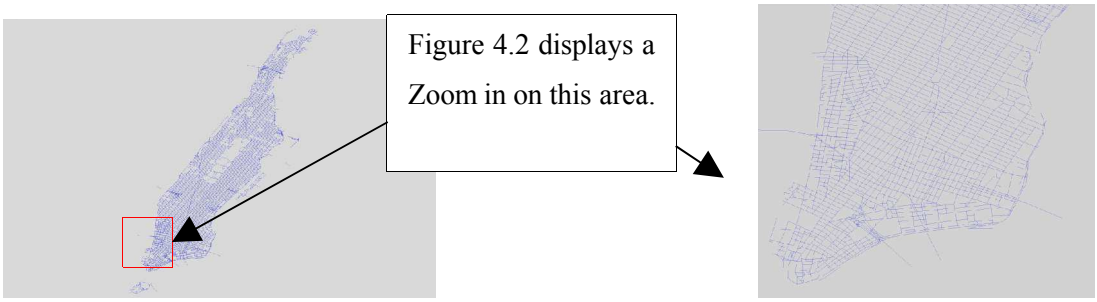
The developed SRF application was based on the SAGESS architecture. This style of architecture is a three-tier architecture. The fist tier, the client communication tier, is an independent tier that allows any user with a wireless device and Internet browser capabilities to connect to the application. The application tier runs the Apache Web server on Linux Red Hat 9 operating system. Apache was selected because it is a free server that can run on many different platforms. The *SQL/XML translator* is implemented using the Oracle XDK for Java component.

In the GIS tier the SRF application access geographical data types from an Oracle 9i database with a spatial cartridge running on Linux Red Hat 9 based platform. Oracle was selected because it is the world leading database in Spatial, XML technologies and has been used in various other commercial and research systems. The SRF core application code was implemented in Oracle PL/SQL and is stored internally as a database stored procedure. The database procedure accepts parameters of the users location coordinates and service criteria. The SRF stored procedure uses a shortest route algorithm based on a link algorithm. The SRF stored procedure will utilise dynamically updated traffic, restaurant and road works tables to dynamically produce the shortest route. The SRF stored procedure will examine roads inside a service region area. The service area is based on the distance from the users current location to the service location multiplied by a geographic factor. This procedure will dramatically speed up route calculations because roads not within the service area are excluded from the search. If no route can be found then the service area is increased, the algorithm works on entire roads so if the start of a motorway is found, the entire motor junction inside the area will be included. The shortest route algorithm takes into consideration road speeds and other outside factors. If the user is very far away from the service, (for example the length of an entire country) then a link based algorithm search can be timely. To resolve this issue knowledge based routes on mid-range points are used. These routes are updated when appropriate traffic information source is received. The execution results of the stored procedure are stored in a user specific object-relational table. This facility allows for added security and base application tables to be made read only, preventing multiple application read/write locks. Database stored procedures add an extra layer of security because users can be granted only execute access on the procedure and not the underlying tables.

SRF Prototype System

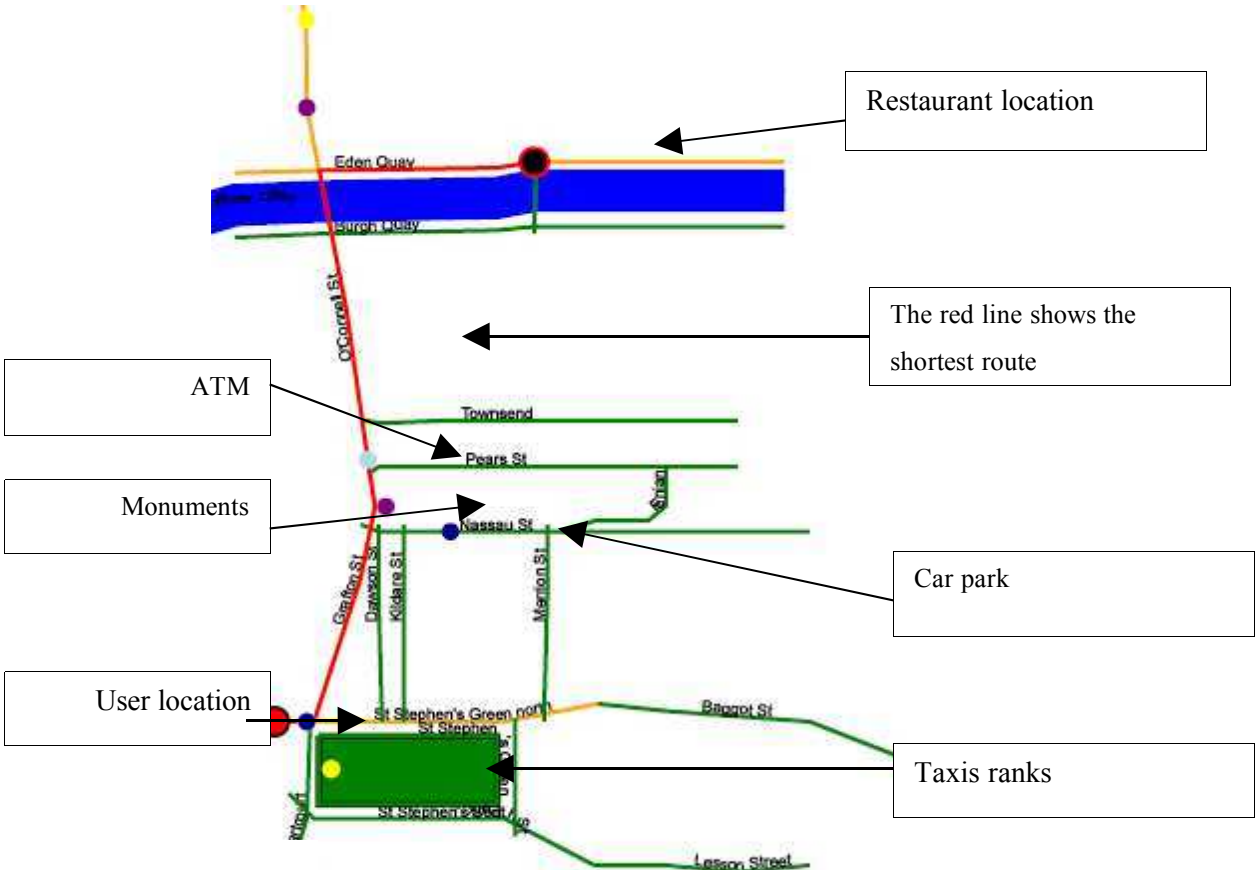
The SRF prototype system was tested on two areas: Dublin City and New York City. The commercial GIS company MapInfo provided the map for New York City and an Oracle

supplied conversion tool was used to convert the map into an Oracle geometry type. The map of Dublin city was created by the author and is based on a non-electronic commercial map of Dublin city. The author created the map of Dublin to demonstrate a solution to the problem of obtaining digitalised map data. The map creation follows the guidelines of cartography map development and database geometric types. These two cities have two completely different city layouts. New York is grind oriented while Dublin like a typical European city with many side streets in no fixed pattern. Figure 4.1 and Figure 4.2 below demonstrate the dynamic SVG map generated from the SRF application of the entire city of New York. The image is generated from 80000 rows stored inside the database. To display the image takes a bit over five seconds on a typical GPRS connection.



SVG New York City ***Figure 4.2:*** ***Zoom in on NY City***

The SRF application implements the most popular of LBS applications, the services guide. The SRF application implements a restaurant guide as it prototype service guide. The user can select a restaurant based on restaurant type and price. below displays the result of a user selecting a "French" type restaurant at the price of €40 and walking as the method of transport.



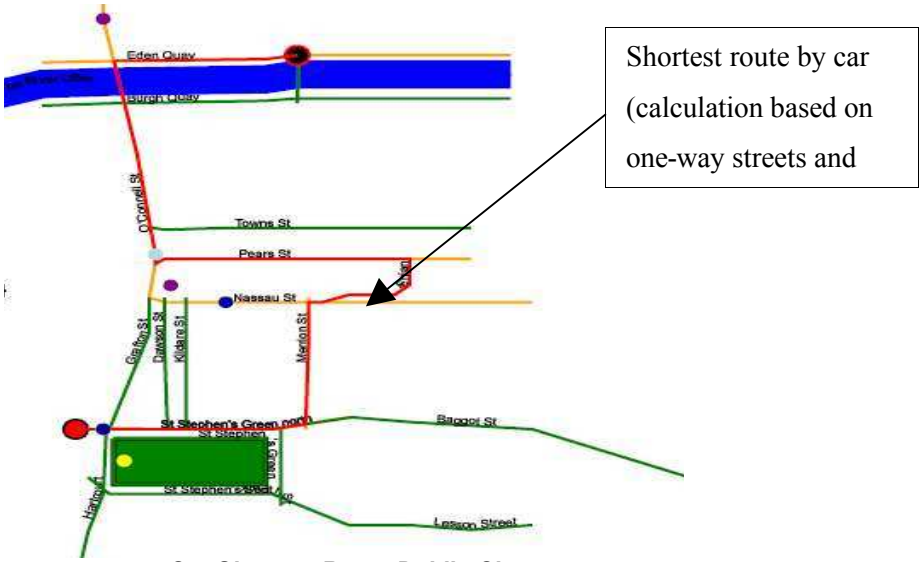
Walking Route Finder Dublin City

The dynamically generated SVG maps offer intelligent map features, not representative in non-electronic maps. These intelligent features include a road text enhancement, route travel details and restaurant Internet address hyperlinks. Road text enhancement displays the name of the road in larger text when the user clicks on a road. This feature is designed for better visibility and to resolve issue relating to text on line support in mobile SVG.

The user can select the nearest restaurant regardless of type. below demonstrates the SRF application ability to deliver user centric maps. The map area displayed in below is relevant to the users' specific queries.

Dublin City Walking Nearest Restaurant

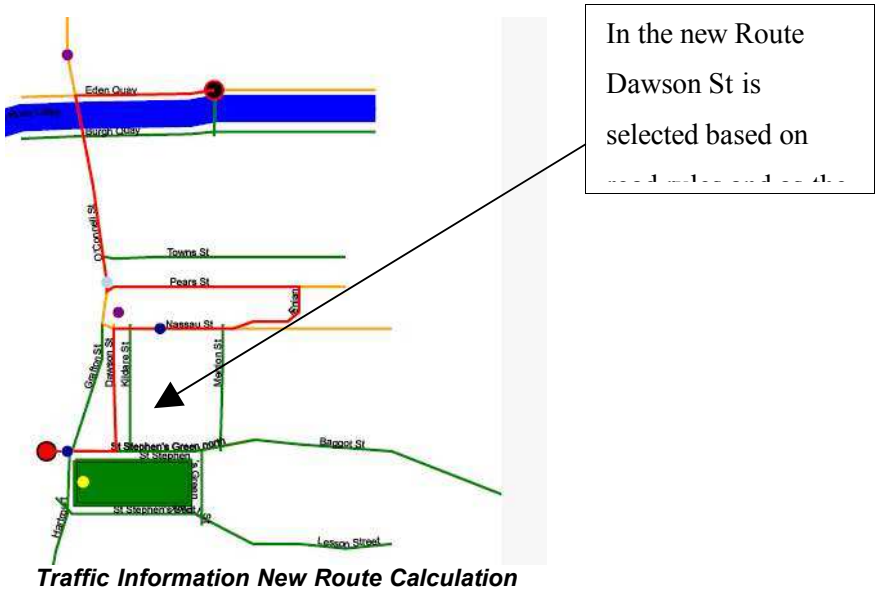
demonstrates the results when a user selects driving as their desired method of transport and "French" as their restaurant type with a menu price of €40. The users current location is the same as the two previous queries. The same XSL for the previous query is used to dynamically generate the map.



Car Shortest Route Dublin City

The dynamically generated map offers all the same capabilities as the Walking XSL in . The SRF application also displays the routing information specific to transport means and route selected. The development time and complexity of the car shortest route module into the SRF application was decreased because it is based on the Walking shortest route template and the core application logic. This demonstrates the power of the SAGESS architecture and the SAGE development strategy, as new application areas and product development is possible based on generic templates.

The SRF application incorporates dynamic outside traffic information to improve on the LBS application result accuracy. below the traffic information feed was received from an outside source (AA road watch, traffic light statistics, News information feeds). The dynamic outside traffic source informs of a 20-minute delay on the Merrion Street. below demonstrates the SRF application ability to adjust this dynamic traffic information to recalculate the shortest route.



Traffic Information New Route Calculation

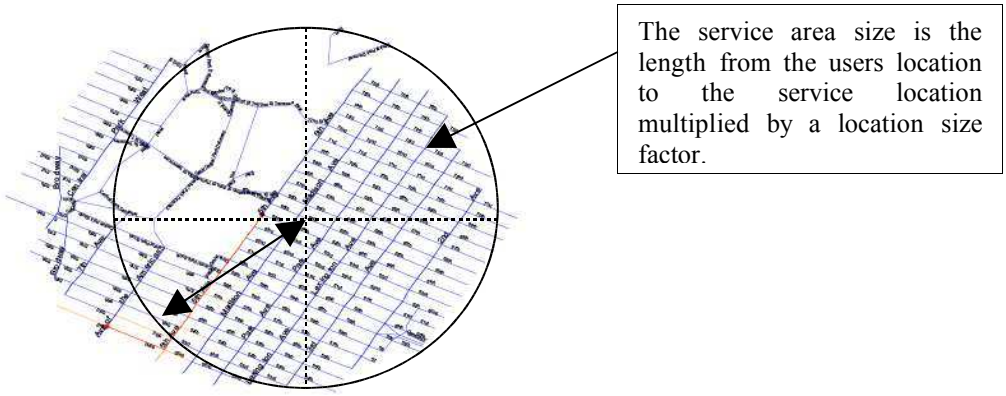
The SRF application was designed with two test areas New York and Dublin. below demonstrates the SRF application on New York City.

The New York City development was based on the previous Dublin City development. The core application LBS Dublin City code required only minor alterations of a New York map table and New York information tables. The New York XSL was developed based on the Dublin City template and requires only to change the ratio number scale. There was only a small requirement of change because the XML dynamically generated document was based on generic table names. The dynamically generated New York SVG maps offer all the same interactive capabilities as the Dublin maps because they are based on the same XSL template. The use of templates and generic XML tags means that a first time LBS developer could have deployed the above LBS application with only two minor changes.

The user service area is the search area to be used to calculate the shortest route. below the entire map of New York is displayed. To search all of New York to find every possible route could take days or weeks and the map delivered would be difficult for the user to search.

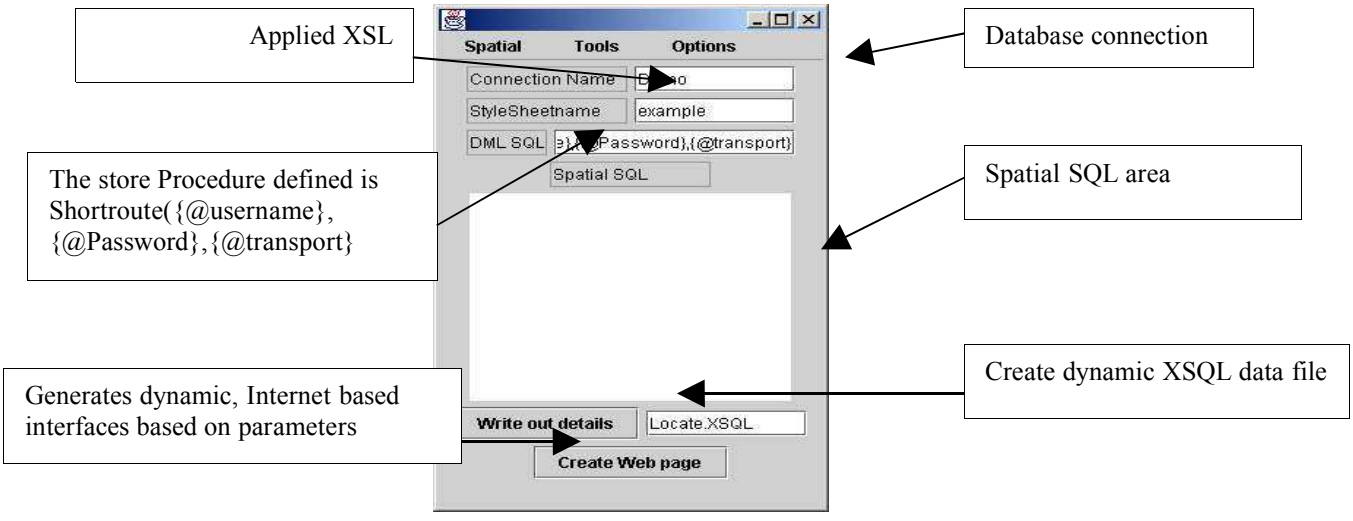
Entire New York Search Area

The SRF application delivers an SVG image of just the relevant service area. below the service area of the query is displayed. The use of a service area significantly decreases the shortest route calculation time.



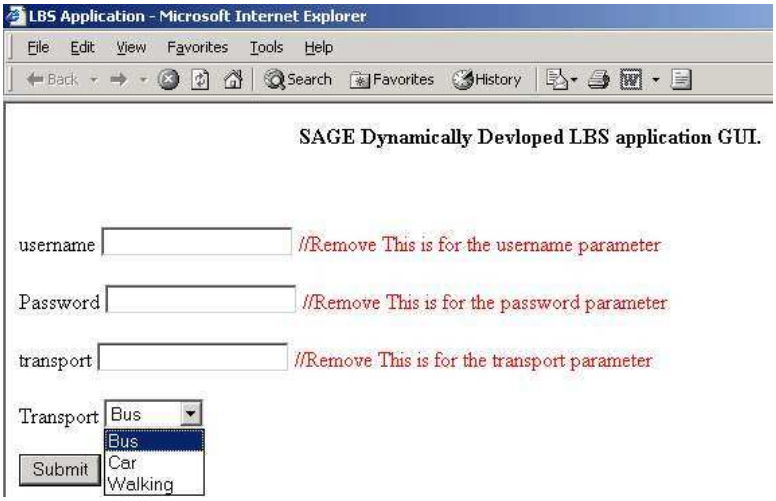
New York Service Area

below demonstrates the SAGE development toolkit.



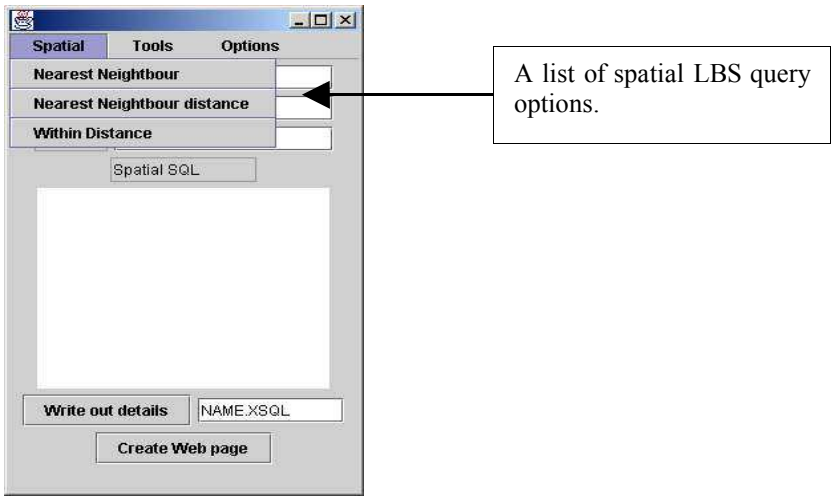
SAGE Toolkit LBS Interface Generation

Three Parameters username, password and transport will be included as input boxes in the new dynamically created Internet page, displayed in below. The key word "transport" generates a drop down list box of common transport means.



Dynamically Generated Internet Page

below demonstrates the spatial LBS query options.



SAGE Toolkit Query Options

Conclusion

An open standard approach to LBS application development will enable LBS developers to deliver LBS applications that can be flexible and scalable to an ever-changing wireless telecommunications market. The SAGESS architecture and SAGE development methodology provides a framework from which LBS developers can deliver the next generation of dynamically generated user specific LBS applications based on industry standards.

REFERENCES

Loehnert E., Wittmann E., Pielmeier J. Sayda F. (2001). PARAMOUNT- Public Safety & Commercial Info-Mobility Applications & Services in the Mountains. 14th International *Technical Meeting of the Satellite Division of The Institute of Navigation ION GPS.*

Forman G & Zahorjan J. (1994). The Challenges of Mobile Computing, *IEEE Computer, Vol.* ***27,*** **No. 4 pp 38 – 47.**

Rui José, & Adriano Moreira, & Filipe Meneses, & Geoff Coulson. (2001). An Open Architecture for Developing Mobile Location-Based Applications over the Internet. *6th IEEE Symposium on Computers and Communications, Hammamet*, Tunisia.

Jin Jing, & Abdelsalam Sumi Helal, & Ahmed Elmagarmid. (1999). Wireless Client/Server Computing for Personal Information Services and Applications, *ACM Computing Surveys*, Vol. 31, No.2.

Dunham M. H. & Helal A. (1995). Mobile Computing and Databases: Anything New?
***SIGMON Record*, Vol. 24, No. 4, pp 5 – 9.**

Albena Mihovska & Jorge M.Pereira. (2001). Location-Based VAS:Killer. *Applications for the Next-Generation Mobile Internet Personal, Indoor and Mobile Radio Communication,* 12th IEEE International Symposium .

John McKeown, & Jane Grimson. (2000). SVG: putting XML in the picture: *Proceedings of XML Europe Paris, France, Graphic Communications Association (GCA).*

Alonso R. & Korth H. F. (1998). Database System Issues in Nomadic Computing,
SIGMON Record, 1998, pp 388 – 392.

Poslad S, Laamanen H., Malaka R., Nick A. Buckle P. and Zipf, A. (2001). CRUMPET: Creation of User-friendly Mobile Services Personalised for Tourism. Proceedings of: 3G 2001 - *Second International Conference on 3G Mobile Communication Technologies.*

Bharghavan, V. & GUPTA, V. **(1997)**. A framework for application adaptation in mobile computing environments. In Proceedings of the 21st *International Computer Software and Applications Conference (COMPSAC '97)*. IEEE Computer Society, New York, NY, 573- 579.

Alexander Zipf. (2002). User-Adaptive Maps for Location-Based Services (LBS) for Tourism. In: K. Woeber, A. Frew, M. Hitz (eds.), Proc. of the *9th Int. Conf. for Information and Communication Technologies in Tourism* ,Innsbruck, Austria.