Published in:
Location Based TeleCartography and Services (Lecture Notes in Geoinformation and Cartography),
G. Gartner, et al. (eds.). Springer-Verlag: Berlin, etc., pp.511-524.

Bata Positioning System - A real time tracking system for the world's largest relay race

B.J. Köbben[1], D. Boekestein[2], S.P. Ekkebus[2], P.G. Uithol[2]

[1] International Institute for Geo-Information Science and Earth Observation, PO Box 6, 7500 AA Enschede, The Netherlands
kobben@itc.nl
[2] BPS team <http://www.batalive.nl; info@batalive.nl>

Abstract

Modern technology makes it possible to develop applications that determine the position of any object. Using affordable off-the-shelf components combined with internet-based applications, we developed the Bata Positioning System (BPS), a real time system for tracking moving objects. It has a focus on accessibility to the user and system scalability. BPS combines the power and flexibility of satellite positioning, mobile telecommunications and open standards compliant internet applications in a unique way, and was first applied for the Batavierenrace, the world's largest relay race.

We will first briefly introduce the Batavierenrace and then explain how the wish to maintain an overview of this race led to the development of the BPS 1.0 prototype. Its basic requirements and general setup as well as the systems architecture will be discussed. It will be explained how further refinement of this architecture led to the current stable version BPS 2.0. We discuss how the BPS 2.0 system principally provides Location Services, but has a great potential to further grow into into a genuine real time Location Based Service (LBS). In the last section, the prototype BPS 3.0 system providing such LBS functionality is presented.

Published in:
Location Based TeleCartography and Services (Lecture Notes in Geoinformation and Cartography),
G. Gartner, et al. (eds.). Springer-Verlag: Berlin, etc., pp.511-524.

The Batavieren relay race

The Batavierenrace ([1]), 'Bata' for short, is a yearly relay-race with participants mainly from the student population of the Netherlands. The route of the Bata covers over 185 km, divided into 25 stages (17 men's stages and 8 women's stages) varying from 3.4 to 11.9 km. At midnight the race starts in the city of Nijmegen and the participants will run throughout the following night and day through the Eastern Netherlands and neighbouring Germany, to arrive about 18 hours later in Enschede. On average over 300 teams participate in this race, and the 2005 edition's 7825 runners earned it the Guinness-book record of 'largest relay race in the world'. For the majority of the participants, the race is just leisure. But there is also a very serious University Competition, in which the best runners battle for the fastest total time.

The history of the race goes back to 1972, when a couple of students from Nijmegen University came back from a similar relay run in Sweden determined to organise such an event in the Netherlands. The resulting first race was run from Nijmegen to Rotterdam, based on the route of the *Batavieren* tribe, which (according to folk myth) came rowing down the river Rhine some 50 years BC. The Batavierenrace still owes its name to that first route, although it was later redirected towards Enschede.

Most teams use minibuses for the transport between check points, where they have to arrive in time to collect the incoming runner and drop off the participant for the next stage. With all these people moving around and the race taking place mostly in a rural area, partly during the night, it is very hard to be aware of the status of the race. Most of the time, the teams themselves as well as other people interested have no idea where the various runners are at any given moment. Having actually experienced these problems themselves, a couple of students started thinking of a solution that could provide participants and spectators with the location of runners in real time during the race. This was the start of the Bata Positioning System (BPS).

BPS: The Bata Positioning System

The development of the system is an ongoing effort by the core BPS team: Three students from the *University of Twente* Telematics BSc/MSc and one from the *Hogeschool of Utrecht* electronic engineering Professional Master programme. This team has been supported by sponsors and several

Published in:
Location Based TeleCartography and Services (Lecture Notes in Geoinformation and Cartography),
G. Gartner, et al. (eds.). Springer-Verlag: Berlin, etc., pp.511-524.

academic supervisors, while some students intertwined this project with their studies, resulting eg. in bachelor theses [2] and [3].

A first version of the BPS was tested in the April 2003 race, and in the following years most parts of the system have been upgraded and refined. In the next paragraphs, the general setup of the current system, which is based on the use of raster geodata, will be presented. Next, there will be a discussion of the next generation BPS, based on vector geodata, and its potential to grow from a Locating Service into a system supporting full–fledged Location Based Services.

Basic requirements and setup

As stated in the introduction, the BPS was designed to track runners during a relay-race. Therefore, the prime requirement is that it can provide real time position information of these participants. This information must be presented to the end-user in such a way that the user is able to get an overview of the runner's locations in (near) real time. The system itself must also be easily adaptable and deployable, able to use existing networks and infrastructure. In this way, it can be easily used in any area, for various types of races, without the need for extra infrastructure.

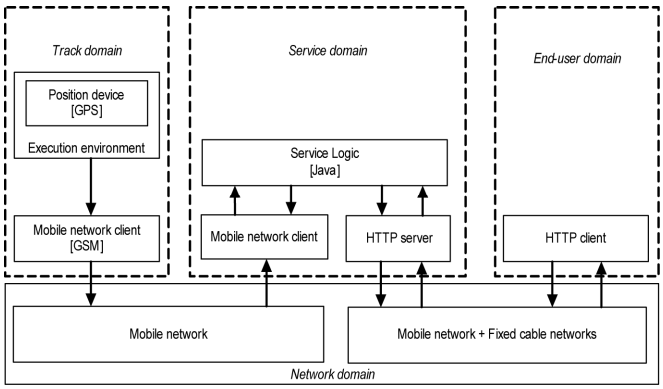

Fig. 1. The four domains of the Bata Positioning System

Four different domains can be distinguished in the BPS setup, as depicted in figure 1. First there is the *track domain*, which provides positioning information about the tracked objects. Each of these tracked objects is an autonomous entity, which uses a push mechanism to transmit its position information to the service domain. The number of objects being tracked by the system must be easily expandable, so the procedure of

Published in:
Location Based TeleCartography and Services (Lecture Notes in Geoinformation and Cartography),
G. Gartner, et al. (eds.). Springer-Verlag: Berlin, etc., pp.511-524.

adding and removing tracked objects is implemented using a plug-and-play mechanism. The positioning device is a simple GPS receiver, which is used by the execution environment to obtain information about its latitude and longitude (on the WGS84 datum) and speed and direction. After the position has been retrieved and some metadata (time, team, runner, etc.) has been added, the execution environment sends the data to the service domain through a mobile network client using the GSM network. Since the existing GSM infrastructure can be considered standardized throughout Europe, this method for transferring data is generally applicable. To transfer the data either SMS, WAP, GPRS or nowadays UMTS can be utilized.

The *service domain* is the central part of the system. Here, the information received from the track domain is parsed by the mobile network client, for use by the service logic, which is implemented in Java. An HTTP server makes the output of the service logic available on the internet.

The HTTP client in the *end-user domain* retrieves the information from the internet and represents the interface to the end-user.

The *network domain* provides the connectivity for the data communication between the domains. Whereas the track domain and the service domain always use the GSM mobile network to communicate, the service domain can be connected to the end-user domain through either a fixed network (dial-in modem, cable-modem, LAN) or any of a number of mobile networks (eg. WAP, GPRS, UMTS, WiFi).

BPS 1.0 – the protoype

A first version of the BPS was tested in the April 2003 race, as a limited pilot, tracking only three participating teams, plus the BPS-team minibus. In this first incarnation the *BPS-boxes* (tracking devices) were rather large and filled with the components shown in figure 2: a commercially available GPS receiver module (1), a standard GSM phone (3), connected through and controlled by a self-built microcontroller (2). Not shown in this picture is the battery power pack. Despite testing beforehand, these power supplies were draining much faster than expected under 'real race' conditions. They consequently had to be replaced during the race.

Published in:
Location Based TeleCartography and Services (Lecture Notes in Geoinformation and Cartography),
G. Gartner, et al. (eds.). Springer-Verlag: Berlin, etc., pp.511-524.

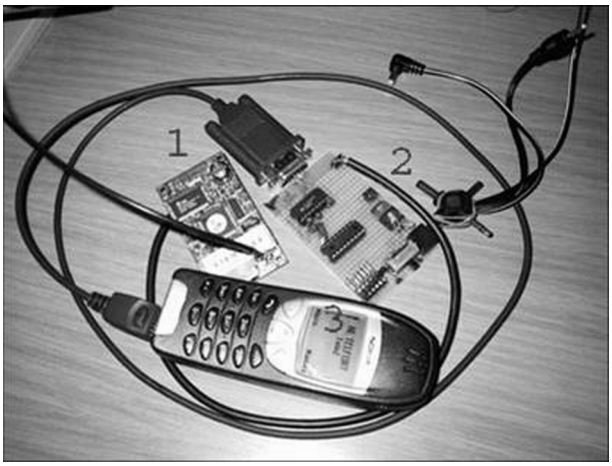

Fig. 2. The component parts of the first version of the BPS box

In this prototype version, only SMS (Short Message System) messages were used to transfer the position data from to the service domain. Although GPRS (General Packet Radio Service) data transfer in principle was available, we chose not to use that. Firstly because the race route is on the border with, and partly crossing into Germany, and therefore the cellular network along the route did at the time not have full un-interrupted GPRS coverage. Secondly, for GPRS we would have had to implement the PPP, IP and UDP protocols in the microcontroller. This is certainly possible, but not within the short time vailable before the 2003 race.

Despite it advantages, there are also some major draw-backs to SMS. Firstly, there is a limited message size and character set (160 7-bits characters) and secondly the price per data unit is very high. The latter was of less concern, since the telecom provider Telfort (http://www.telfort.nl) was sponsoring the pilot by covering these costs.

In figure 3, a more detailed view of the architecture of the *service domain* is depicted. The *position database* is the central part of the service domain. In the pilot, we used the popular open source database server MySQL ([4]). Unfortunately, it transpired that we required the use of sub-queries, which were only supported in the then latest version, MySQL 4.1.0 alpha. Using an alpha version for anything other then testing is of course never advisable, and it would turn out to be a bad choice indeed. During the race whenever database usage became more intensive, and as soon as the number of position entries became more than a couple of thousand, the database server locked up completely. The only solution available was cleaning up the table contents at regular intervals.

Published in:
Location Based TeleCartography and Services (Lecture Notes in Geoinformation and Cartography),
G. Gartner, et al. (eds.). Springer-Verlag: Berlin, etc., pp.511-524.

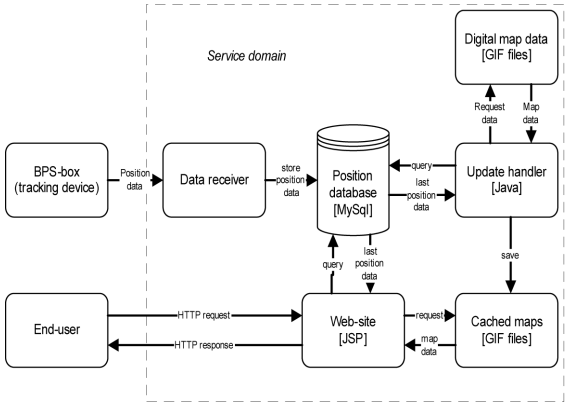

Fig. 3. Service domain architecture of BPS 1.0

The goal of BPS is to give anyone interested an insight in the progress of the race and for that just the raw position information is not nearly enough: No-one will be helped much by lists of latitude/longitudes… Therefore we needed to project these positions onto a map. Figure 4 shows a screen shot of the interface with the zoomed map. The team selected by the user is centered in the middle of the map, represented by an arrow. The angle of this arrow is defined by the travel direction retrieved from the position database. Any other team whose position is within the viewed area is shown as a red dot. Using the input list in the bottom of the screen, a user can choose to center the view on another team, or go to an overview map.

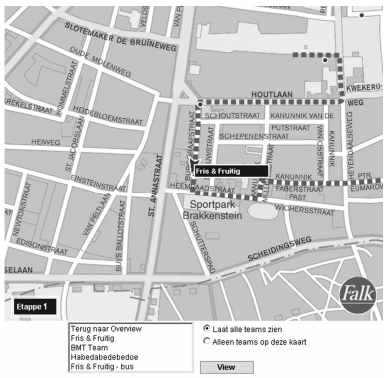

Fig. 4. User interface of the BPS 1.0 pilot

Published in:
Location Based TeleCartography and Services (Lecture Notes in Geoinformation and Cartography),
G. Gartner, et al. (eds.). Springer-Verlag: Berlin, etc., pp.511-524.

The maps used for displaying the position of the runners where made available by another sponsor (Wegener Falk, now Falkplan-Andes BV, http://www.falk.nl/). These were raster files of their commercial street maps, tiled into convenient chunks and stored in GIF format on the server. From these base files, the *update handler* constantly creates a cache of map fragments centered around the team's positions. This caching improves the performance of the *web site* JSP component (Java Server Pages), making the system more scalable. When an end-user requests the current positions on the website, the JSP will retrieve the latest positions and other relevant information from the position database and use that to determine which cached map data to retrieve and how to show it in the user interface for the *end-user*.

Despite the teething troubles mentioned, the general setup of the system, tracking runners and offering the collected information on the internet in real time, worked satisfactory. It was certainly promising enough to further develop a more reliable working system.

BPS 2.0 – the current system

The BPS 2.0 system as it is available at the time of writing is considered a stable version. It has been tested at the April 2004 Batavierenrace and further refined during subsequent events, taking place in different parts of The Netherlands. Based on the experiences with the protoype, it was decided to leave the general setup of the system unchanged, and concentrate on making it more robust and user-friendly.

For the *track domain*, a much smaller BPS-box has been developed (figure 5). With a size of only 12 x 6.5 x 3.6 cm and a weight of 200 g, it can comfortably be carried by a runner. When the internal battery is fully loaded, it will function for approximately 8 hrs. Additionally, external battery packs or other sources of energy (such as a 12 V car outlet) can make for an almost endless battery life. Instead of the standard GSM phone there' s now a small on-board component and the hand-built microcontroller was traded in for a commercially available one with embedded Linux (eCos, an open source, real time operating system intended for embedded applications). This allows for flexible, fast and updateable controller logic.

Published in:
Location Based TeleCartography and Services (Lecture Notes in Geoinformation and Cartography),
G. Gartner, et al. (eds.). Springer-Verlag: Berlin, etc., pp.511-524.

Fig. 5. BPS 2.0 tracking boxes (12 x 6.5 x 3.6 cm). On the one in front the connection for the battery loader or the optional external power source can be seen. On top of the other box, the GPS antenna is visible.

GPRS is used, instead of SMS, as the primary means of data transfer, making more frequent updates feasible, as the price per data unit is much lower. Also there are no restrictions on the data representation: one can use any message format, with any character encoding. In case GPRS connectivity fails, the system can fall back on SMS until GPRS coverage is restored.

The position database now is implemented on the PostgreSQL database server ([5]), even though MySQL now does support sub-queries in its stable versions. The main reason for the switch was to provide for the future deployment of a spatially enabled database (see next section).

The client-side of the system was enhanced also. The website now can offer multiple zoom levels and provides the end-user with more information, during as well as after the race, such as replays of the tracking and speed diagrams (as shown in figure 6). Furthermore, at the highest zoom level not only maps are provided, but also aerial photographs, courtesy of the sponsor Aerodata International Surveys (http://aerodata-surveys.com/).

Published in:
Location Based TeleCartography and Services (Lecture Notes in Geoinformation and Cartography),
G. Gartner, et al. (eds.). Springer-Verlag: Berlin, etc., pp.511-524.

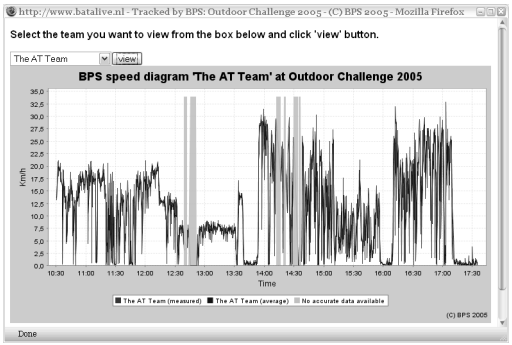

Fig. 6. Example of a speed diagram, generated during the Outdoor Challenge 2005, an orientation race with a combination of running, canoeing, cycling and kick-biking.

The BPS 2.0 system provides the real time position of the runners during the race and displays their positions on a map. As such, it is described most accurately as a Location Service (LS). Using the type of data we accumulate, it is also possible to combine that LS with all kinds of other spatial data and services, giving additional feedback to the runners, their teams and the spectators. As an example, the system could warn the runners when they are off-track, or let them know how close the nearest competitor is, or how to get to the nearest hospital in case of an accident. Offering this type of services, the system would turn into a genuine Location Based Service (LBS).

BPS 3.0 prototype – Development of a real time LBS

There are many definitions of an LBS around, a common one being LBSs are information services accessible with mobile devices through the mobile network and utilizing the ability to make use of the location of the mobile device ([6]). The BPS system is theoretically able to provide the Location Based Services mentioned above. In what manner this can best be implemented was investigated as part of the research for the Bachelor Thesis in [3], and based on this a prototype has been developed. This prototype uses the same basic setup as the BPS 2.0 system, but it differs considerably in two aspects:

1. the geographical data is stored not in (raster) *map* form, but as geographic vector *data* in a spatially enabled database, and

Published in:
Location Based TeleCartography and Services (Lecture Notes in Geoinformation and Cartography),
G. Gartner, et al. (eds.). Springer-Verlag: Berlin, etc., pp.511-524.

2. this data is depicted in a user interface based on the Scalable Vector Graphics (SVG) format instead of raster GIF images.

The prototype is built using an n-tiered distributed design, with the possibility to spread tiers across servers, in order to improve scalability and performance, as shown in figure 7.

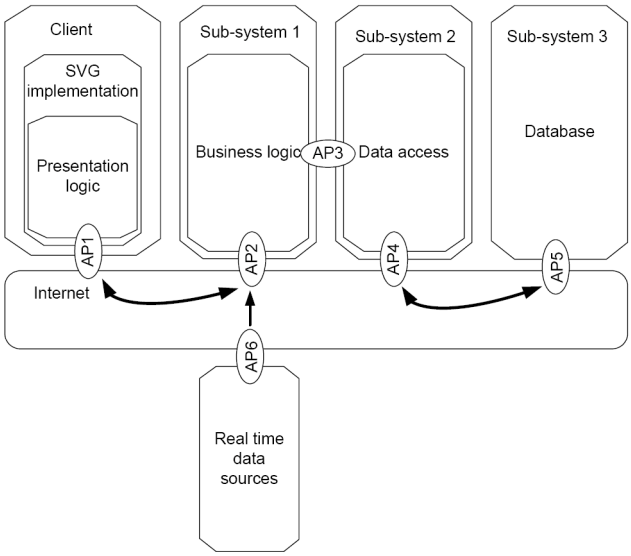

Fig. 7. The tiered architecure of the BPS 3.0 prototype (AP = Access Protocol).

The *Database Tier* is where the actual data storage is performed. The current BPS 2.0 stores in its database only data of the runners' locations. The topographic information is in the GIF map files but only implicitly, for viewing purposes, as the system itself has no knowledge of the spatial components of these maps (other than the georeferencing). One obvious problem with the current non-spatial database is that the system is fixed on data and maps projected in the Dutch national grid (Rijksdriehoeksstelsel). A conversion of the system to use non-Dutch maps would be possible, but it would have to be redone for every new projection. More importantly, the current system lacks support for *spatial queries*. Because of that, it is impossible to ask questions like "find all runners within a certain area", or "how far ahead am I of my nearest competitor?" to the database.

If one uses a database to serve as the backend of a geospatial service such as BPS, the most obvious choice is a so-called *spatially enabled* database, that is one that has special data types, functions and operators to

Published in:
Location Based TeleCartography and Services (Lecture Notes in Geoinformation and Cartography),
G. Gartner, et al. (eds.). Springer-Verlag: Berlin, etc., pp.511-524.

deal with real-world geometries. Such spatial extensions, or spatial cartridges as they are sometimes called, provide access to the geometry through an SQL interface. Open standards for this interface have been set by the Open Geospatial Consortium (OGC, [7]). There are several databases with OGC-compliant spatial extensions. PostgreSQL (when using the PostGIS plug-in [8]) and MySQL are open source examples thereof. PostgreSQL/PostGIS offers a more mature and complete implementation, and furthermore MySQL spatial functionality is only implemented for the MyISAM table type. MyISAM does not support more advanced database features such as foreign keys, triggers and views, and especially lacks *transaction* capabilities, which are essential in a multi-user system such as BPS.

The database serves multiple roles in the system: Firstly, it stores and retrieves spatial data of the topography, which is then used to generate SVG maps. PostgreSQL/PostGIS enables generation (and caching) of the chosen output format SVG (see below) natively in the database. This saves a lot of conversion time later on, when hundreds of requests are made for the same piece of data.

Secondly, it contains functionality related to the processing and storage of the real time positioning data. By deploying custom-written internal functions, we have been able to cut down the number of separate queries necessary for complicated actions like setting up a new participant. This greatly reduces the time needed for client–server communication.

The system accesses the database through the *Data Access Tier*. This interface to the data tier handles all data input and output. Its main task is to keep track of all data sources and provide access to these through a Connection Pool. The mechanism permanently keeps a number of connections to the database open, ready for immediate use and avoiding the continuous and time consuming opening and closing of connections. We've also implemented a form of priority queuing, as some data requests are more important than others, and are thus given precedence when resources are scarce.

The *Business Logic Tier* handles incoming requests, by figuring out exactly what data is needed to answer the request, making sure the requested data is received, and transforming it into a valid response.

The communication between the tiers is realised using *Acces Protocols* (AP*n* in figure 7). These follow the Web Services paradigm, being regular HTTP requests, using a pre-defined set of GET variables. Requests are received by one servlet that is a part of our business logic layer (with a URL in a format of http://serveraddress/Request?request=TheRequest). This servlet then distributes the request to the class responsible for answering that specific type of request. The available requests are

Published in:
Location Based TeleCartography and Services (Lecture Notes in Geoinformation and Cartography),
G. Gartner, et al. (eds.). Springer-Verlag: Berlin, etc., pp.511-524.

RealTime, *GetMap*, *GetParticipants* and *GetNearest*. In these interfaces, we implement selected features from OGC's Web Map Services specification (WMS, see [9]) and use its nomenclature and, if possible, semantics for parameter names and behaviour where applicable. Normally, all requests to the server are made by client tier, but it is possible to use these interfaces by calling them directly from any application (eg. any web browser).

A special case among these interfaces is the *RealTime* interface, since it does not request any data from the server, but rather submits new data of the runners' position. The original plan was to try to comply with the OGC Open Location Services specification ([10]). However, on closer inspection we noticed that the fundamental setup of this standard is that a client is unaware of its location, and asks the server to determine its position. This is the reverse of how BPS works, where the clients (in this case the tracking boxes) know exactly where they are, and want to transmit this information to the server. Therefore we decided to use WMS style parameters for this interface instead.

The *GetMap* request is used to retrieve map data in vector format from a database. Which databases are available, and how and from which tables this data is gathered is defined in a configuration file. The GetParticipants interface generates a listing of the latest available data for participants that have a database entry. This includes data such as a participant's position, name, stage and speed.The GetNearest interface can be used to locate the nearest hospital, first aid point, competitor or other object relative to a given participant. It returns the Euclidian distance to these objects.

The *Presentation Tier* is physically on the client's machine. This component is the end user view of the system and includes the Graphical User Interface (GUI). With the original BPS system, we did put almost no requirements on the client browser and therefore the obvious choice was to use raster GIF images. This enables all users to see the maps without the need for additional plug-ins. But using raster means maps are always transported in full over the Internet, and requires the browser to refresh all the map data regularly, even if there is no change in the runner's location and no new map data is required. For BPS 3.0 we chose to use Scalable Vector Graphics (SVG, see [11]), for both the maps and the GUI itself. Using vector on the client allows the creation of a better-looking and more responsive GUI. Examples of this are the use of anti-aliasing on the client, and the loading of new map data in the background without the need to redraw the whole user interface.

In the end the vector data has to be rendered to the screen of course, putting some additional requirements on the client. In most cases, some

Published in:
Location Based TeleCartography and Services (Lecture Notes in Geoinformation and Cartography),
G. Gartner, et al. (eds.). Springer-Verlag: Berlin, etc., pp.511-524.

sort of SVG plug-in is needed, although browsers with native SVG rendering are starting to appear (eg. FireFox 1.5 and Opera 9).

For the client architecture, we used the basic functionality of an existing application framework from CARTO:NET ([12]) and modified it to fulfill our needs. The part we had to add was the dynamical loading of the map data and the participant information. To have the client issue the requests to the bussiness logic, we've used the getURL() function that is currently only implemented in the Adobe SVG plugin, but is also included in the upcoming SVG 1.2 recommendation. The output of all three *Get* requests mentioned above are snippets of SVG vector data, which are parsed into the client SVG.

The end-user has the possibility to select a participant and zoom in on his location. One can then can choose to track the participant automatically or to see where the nearest competitor, hospital, ectetera is located. The participant's information is refreshed at a chosen interval and his position updated on the map. When the participant goes out of the current map frame, new map data will be retrieved and the map is panned accordingly.

In order to evaluate the merits and drawbacks of the prototype, we compared it to the current BPS 2.0 system. With regard to the *performance*, the first load of the prototype is slightly slower, since multiple requests need to be made to the system, but after this it is significantly faster at handling user requests. In terms of *scalability*, the prototype is comparable to the current BPS. Both can be spread over a large number of servers, using Tomcat's built-in load-balancing features and replication for the database. The *extensibility* of the prototype is better, due to a much better separation between presentation and business logic, and the way in which requests are structured. The prototype's new *functionalities* will give the end-user more interaction and enables all kinds of spatial queries which are not possible with BPS 2.0. The use of SVG in the client makes the system less platform-independent on the one hand, but generally the use of vector data in both server and client gives a lot of new possibilities, which we haven't explored fully yet.

Conclusion

The BPS system combines off-the-shelf components and combines them with open standards compliant internet applications into a highly efficient and powerful application. The current BPS 2.0 is a low cost and scalable system, making it possible to track many objects simultaneously and

Published in:
Location Based TeleCartography and Services (Lecture Notes in Geoinformation and Cartography),
G. Gartner, et al. (eds.). Springer-Verlag: Berlin, etc., pp.511-524.

display their positions on the Internet in real time. It has proven its useability in several use cases under real-world conditions.

In its current BPS 2.0 version, it can be argued that it principally provides Location Services, but it has a great potential for Location Based Service (LBS). The prototype BPS 3.0 that we presented still needs some fine-tuning and can be improved in several ways, but already we think its proved a solid basis for further development of a flexible, multi-purpose, genuine real time LBS.

References

[1] Batavierenrace website. http://www.batavierenrace.nl/english/ (last accessed: February 2006).
[2] Ekkebus, S. P. (2003): Realtime location based positioning system - a requirement study and design. Bachelor thesis. Enschede: University of Twente, EWI. 63 p.
[3] Boekestein, D. & P. G. Uithol (2005): Location based services using database driven vector based graphics. Bachelor thesis. Enschede: University of Twente, EWI. 72 p.
[4] MySQL AB website. http://www.mysql.com/ (last accessed: February 2006).
[5] PostgreSQL website. http://www.postgresql.org/ (last accessed: February 2006).
[6] Virrantaus, K., J. Markkula, A. Garmash & Y.V.Terziyan (2001): Developing GIS-Supported Location-Based Services. Proceedings of WGIS 2001 – First International Workshop on Web and Wireless Geographical Information Systems. Kyoto. pp. 423–432.
[7] Open Geospatial Consortium: OGC Home Page. http://www.opengeospatial.org/ (last accessed: February 2006).
[8] Refractions PostGIS website. http://postgis.refractions.net (last accessed: February 2006).
[9] Open Geospatial Consortium Inc. (2002): Web Map Service Implementation Specification (1.1.1) OGC 01-068r3, J.d.L. Beaujardière (ed.).
[10] Open Geospatial Consortium Inc. (2004): OpenGIS Location Services (OpenLS): Core Services (1.0) OGC 03-006r3, M. Mabrouk (ed.).
[11] World Wide Web Consortium (W3C): Scalable Vector Graphics (SVG) 1.1 Specification. http://www.w3.org/TR/SVG11/ (last accessed: December 2005).
[12] carto:net site. http://www.carto.net/ (last accessed: March 2006).