Geodata-based Alerting Service (GAS): A Conceptual Framework to Create Notifications Based on Spatial and Temporal Data

Martin Sudmanns[1], Sebastian Cadus[2] and Mona Bartling[1]

[1]Z_GIS, University of Salzburg, Austria · martin.sudmanns@stud.sbg.ac.at
[2]Research Studios Austria − Studio iSpace, Salzburg/Austria

Full paper double blind review

Abstract

Due to the ever-increasing number of sensors, Web services, and Web feeds, an outstanding range of information and data sources exists. However, at present time, it is rather inconvenient for a user to receive information, which matches not only thematic, but also user-defined temporal and even spatial criteria. As an example, a person might be interested if the weather conditions in the northern part of Lake Garda are suitable for wind surfing in the afternoon, and thus wants to be informed as soon as chosen criteria are met. Therefore, it is the aim of this paper to present a conceptual framework of a Geodata-based Alerting Service (GAS), which matches data retrieved from a Web service against user defined parameters, and – in case these criteria are met – creates an event and sends off a notification. The challenge in this context is to develop a concept as lean as possible, exclusively using existing and common Web technologies, while at the same time handling various data formats and Web services. To foster the acceptance and broad application of the framework, different user requirements, both from the perspective of a service receiver and as a service provider need to be identified and accounted for. Among other things, this requires the development of a graphical user interface, the exclusive usage of open source software, as well as a lean, performant and well documented framework architecture. As a result, a conceptual framework is provided, consisting of different components and exemplary data interfaces. For a broad distribution of the framework, and to foster further development, the framework will be provided for general use via Web-based repositories, once implemented.

1 Introduction

Today, an ever increasing number of geographic data sources and Web services exist, providing a vast amount of data for almost any location on the earth's surface. Although this development provides users with an unprecedented range of information sources, it comes along with an increasing difficulty and effort to retrieve the information of interest. This is especially true for information that refers to a particular area, deviating from administrative units. Thus, to date it is common to refine information based on keywords, semantics and numerical parameters: research by the authors suggests that using geographic space as an additional filtering parameter is not very common outside of typical GIS applications, despite the fact that more than 80% of all available data can be related to space (OHIO

GI_Forum – Journal for Geographic Information Science, 1-2015.
 ISBN 978-3-87907-558-4.
 ISSN 2308-1708, doi:10.1553/giscience2015s6.

DEPARTMENT OF ADMINISTRATIVE SERVICES 2011, WILLIAMS 1987). One of the reasons for neglecting the spatial component could be that data and service providers hardly offer geographic filtering options with their services. A tailored framework for filtering data based on geographic areas offers many advantages: by allowing its user to subscribe to a Web resource, to define a geographic extent and other threshold parameters, refined individual results can be obtained. The user receives a notification of the event via email or SMS. To exemplify this advantage, a person might be interested if the weather conditions in the northern part of Lake Garda are suitable for wind surfing in the afternoon, and thus wants to be informed if the wind speed exceeds 50 km/h between 1 and 5 pm, while the wave height is less than 5 m. As the situation currently stands, the person would visit an appropriate weather Web site, chose all parameters, define one or two weather stations in the area nearby, manually check the conditions and repeat this process several times. In contrast to this, the framework at hand allows the operator of the weather Web site to register his service in the GAS framework (as service provider). A user interested in weather information may subscribe to this Web service through the GAS framework (as a service receiver), define geographic and other parameters, and receive an email or SMS in case the weather data matches his/her predefined parameters.

Possible application fields of the framework are early warning systems, risk management, public safety or tourism. There are also numerous (non-GIS-specialised) organisations and businesses that could be interested in providing their (Web) services through the framework at hand, such as news magazines or health organisations. Target groups that might be particularly interested in receiving filtered notifications comprise – amongst others – researchers in the area of biology (e.g. animal tracking, ornithology) or recreational athletes (e.g. surfer, pilot, paraglider).

It is the purpose of this paper to demonstrate a generic approach for addressing the problem of geographic filtering and event-based notification. First, existing approaches and methods that are related to the issue described are presented. Second, the chosen methodology of the approach is explained. This includes an overview of different demands that potential users (both as service providers and as service receivers) might have, and thus describes the challenges that need to be handled by the approach. Third, the architecture of the conceptual framework, its functionality, and an illustration of the UI are presented. Moreover, a workflow is shown, describing the typical procedure that is passed from the moment that data is received from an external service to the actual user notification. The article ends with a discussion of the results, and an outlook for the future, covering possible extensions and improvements.

2 Why Another Approach?

SAMET et. al (2014) present a similar approach for aggregation and visualisation of news on a map, which are collected by Web crawlers. This system is called NewsStand and shall facilitate “reading news with maps”, as well as the exploration of news, including their spatial and temporal properties. A Web based user interface (UI) provides detailed access to the news. It also provides querying and filtering capabilities of temporal, textural and spatial criteria. However, since the UI of NewsStand is very advanced, the usage might be rather difficult for users without GIS experience, who quickly and easily want to subscribe

to notifications. Furthermore, the subscription for individual notifications currently seems not to be supported by NewsStand.

Complex event processing (CEP) is a method for filtering event data. According to ECKERT & BRY (2009), CEP evaluates incoming event data of multiple Web services based on stored queries, and identifies accordance between the events and the query parameters. After ROBINS (2010, 1), an event may be defined as "an object that is a record of an activity in a system". These activities encompass properties such as time, causality, and aggregation. Based on these attributes, multiple events may be related to each other. Therefore, ROBINS (2010, 1) rightly points out that "CEP can be regarded as a service that receives and matches lower-level events and generates higher-level events". Various approaches to implement CEP have been developed, including SASE and Esper (GYLLSTROM et. al 2006).

The OGC has specified the Web Notification Service (WNS) as a model for server-client communication, especially supporting the use of sensors (OPEN GEOSPATIAL CONSORTIUM 2006). It was designed for delivering messages or alerts from Sensor Planning Services.

However, research of the authors suggests that the extent and complexity of CEP and WNS may in many cases exceed the demands of users, who look for a simple and lean framework to set up and configure notification services. Also, as ECKERT & BRY (2009) have indicated that major application areas for CEP encompass business activity monitoring, sensor networks, and market data. On this basis it may be inferred that the CEP approach does not predominantly focus on geolocated data, and thus pursues a different goal. Current approaches are rather difficult to establish and apply with common Web technologies, and hence require additional software or knowledge. It is the opinion of the authors that this is one of the reasons for the highly specialised application domain of these systems. Therefore, another approach without extensive overhead is suggested, as shown in figure 1.

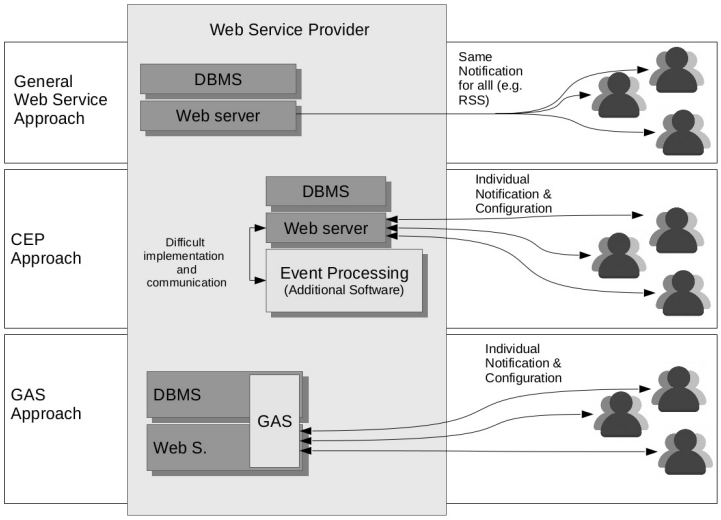

Fig. 1: Advantages of the GAS approach over other state of the art approaches (authors' design)

3 Methodology

3.1 User Demands

In order to correctly capture the demands of potential user groups and thus choose a suitable approach to account for these, two types of users of this framework need to be distinguished: the user as a receiver of notifications on the one hand, who interacts with the framework via a UI; and the user as a service provider on the other hand, who is directly concerned with the setup and configuration of the framework, being able to integrate Web services. This interaction is illustrated in figure 2.

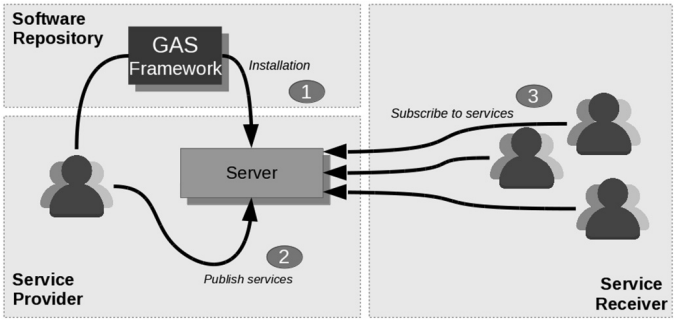

Fig. 2: Differentiation between two types of user groups of the GAS framework (authors' design)

3.1.1 Service Receiver

The framework should comprise an easy to use UI, allowing the user to set up a user account, configure his/her personal data, and manage subscribed Web services (e.g. activetion, deactivation, filter parameters, etc.). Therefore, two basic settings are required: account-based and service-based options. Account-based options imply personal login data as well as the media for transmission, i.e. the way a user wants to receive notifications. This may differ, depending on personal preferences: a user may either want to be notified on a mobile device, a desktop computer, or may even want to further process the received event data on an automated basis. Service-based options on the other hand include several (geospatial) parameters, including thresholds on measured values and temporal restrictions. Five parameter categories were identified to account for different areas of application. These include:

- the area of interest (definition of a single or multiple area(s) for which notifications shall be received),
- a time zone (to facilitate correct time calculations),
- a time span (which determines the temporal period during which an occurrence has to happen in order to be registered as an event),
- the duration of the occurrence (defines how long a status of an occurrence needs to persist to be transmitted; as an example, this might be of interest for animal observation, to make sure that an observed animal does not just pass an area but pauses at a place for a particular time), and
- the frequency of notification (allows to limit the number of notifications, in case a threshold is exceeded several times; for example, an animal may repeatedly enter and leave an area).

3.1.2 Service Provider

Different requirements can be identified for the users who want to install and set up the framework on their server in order to provide their services. A rather important aspect in this regard might be the interoperability of the system with various formats of Web services. Third-party Web services with varying data sources and formats must be easy to integrate. Thus, generic interfaces are required. Moreover, the framework itself must be fully integrable into the architecture of an existing system. To exemplify this, a news magazine may want to integrate the framework into their existing news feed service, making use of pre-existing user accounts. This demands an easy way to modify and extend the framework and therefore requires an independence of commercial products such as ESRI ArcGIS or Oracle database software. Last, the framework should not impose too large burdens on hardware capacities and computing power.

3.2 Challenges

The overall goal of this work as well as the demands described from the users' perspectives come along with numerous challenges regarding the architecture and the database model. An overview of these demands is shown in figure 3.

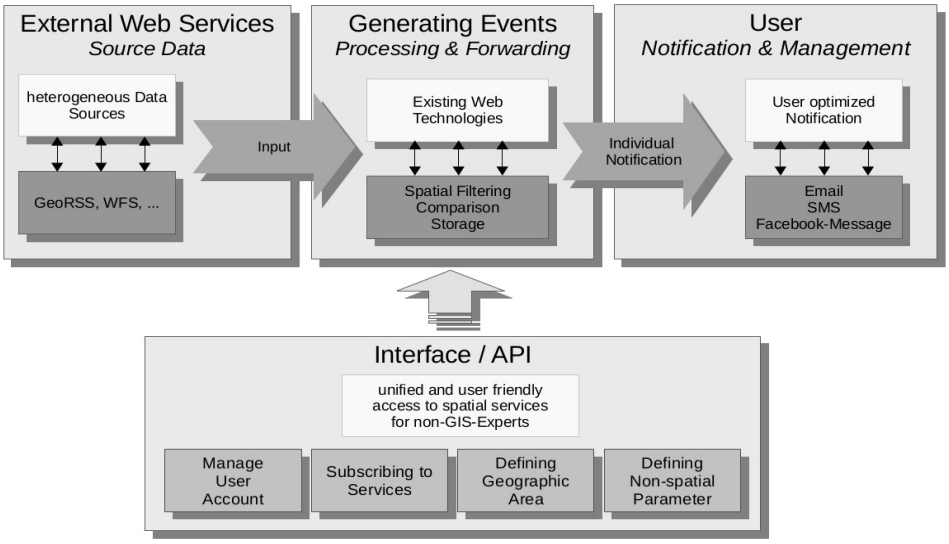

Fig. 3: Overview of the demands on the notification service (authors' design)

An important part of the architecture is a Web-based graphical UI, which is required to facilitate a comfortable way for the service receiver to manage account-based and service-based settings. Users do not need to install additional software on their computer, and are able to access the framework from different devices with their browser. Further, the architecture of the framework should be designed to be able to receive and integrate data from input sources, query the user data, and send out individual notifications as quickly as possible. In addition, the described expectations require the exclusive use of open-source soft-

ware for the development of the framework, along with a reproducible and comprehensible documentation to facilitate a simple system integration, as well as individual adaption and extension. In the end, it is also necessary to ensure a slim and performant software framework, which consists of an efficient, properly planned system structure.

As a further challenge, a data model needs to be designed, which is able to cope with account-based and service-based user parameters, as well as with the actual data of the service (figure 3). The designing of the model should focus on the comparison of the incoming data with parameters defined by the individual user in order to send a notification. Moreover, from the point of view of a service provider, the heterogeneity of Web services and resources (both with respect to formats and types) needs to be addressed. Common formats of source data may comprise different vector formats (e.g. GeoRSS, GeoJSON, XML, WFS-T, etc.) or raster formats (e.g. satellite data products, weather forecast). Possible service types on the other hand encompass push- and pull-services. While former services (e.g. GeoRSS) can be directly received and integrated, the latter services (such as REST) are retrieved either by time controlled queries or user defined requests. For all these cases, suitable connectors are required to transform the data, harmonise it, and store it. Information that needs to be handled in this respect includes coordinates (or other spatial references), a timestamp, the actual information (or measured values), as well as multiple versions for changes in the source data. The latter refers to the fact that data may change after their first occurrence (e.g. forecasts) and thus the version number needs to increase.

3.3 Components of the Notification Service

Based on the demands and challenges highlighted, four main components of the framework can be derived. First, this comprises a graphical user interface (GUI), providing the service receivers with visual support in managing their services. This should allow for subscribing to/unsubscribing from services, defining threshold parameters, and delineating areas of interest, either by means of a drawing tool on a map, or by using a transactional Web Feature Service (WFS-T) of an external tool. Also, while being logged in, the user should immediately be notified of an event through the GUI. Secondly, a spatial database is required, which is capable of storing user data and service data. Additionally, the database should be able to compare the data. Not only in terms of parameter values, but also with respect to spatial intersections. The intersection may be conducted directly in the database; respective methods to store and compare geospatial and non-geospatial data already exist (e.g. in PostGIS). Another advantage of the database approach is that database queries are generally highly efficient because of tuning strategies of the system (e.g. caching) and tuning strategies of the database (e.g. indexes) (SHASHA & BONNET 2003). Hence, databases provide extensive possibilities for improving throughput and performance tuning, depending on the type of queries and hardware. Thirdly, there has to be a component, which accounts for the heterogeneity of third-party Web services, and thus offers an appropriate interface to connect these services to the framework while at the same time assuring a harmonisation of the service data. Fourthly, the framework needs to include an alerting component, which is responsible for the notification of the service receiver in case the input data meets predefined parameters. Optional components and elements for further extensions could include a dedicated mobile application, fuzzy logic or even CEP.

4 Results

The following section consists of four parts. First, a typical event handling procedure (from the data input to the notification of the user) is described. Next, the architecture of an exemplary implementation of the framework is presented, pointing out the components and respective interfaces. In addition, the data model and the composition and functionality of the GUI are explained.

A typical processing chain for incoming data is shown in figure 4. In a first step, data is received from a Web service through the Internet. This data may be of different formats (e.g. XML, JSON, etc.). In a following step, this data is harmonised to transform the varying data formats into a unified format. The unified format is required to store the data in a relational database. As explained in section 2 (Challenges), the format includes the extracted id, coordinates, numerical parameter sets, etc. The transformation is done by a translator (Python script) that has to be provided for each type of input data separately. Once the data is in a unified format, it is transmitted to an importer (Python script), which provides the matching interface. The importer connects to the database and stores the new data in it (SQL). If the data are stored successfully, the importer sends a signal to the controller. Now it is up to the controller to build a query in order to compare the data with the user defined parameters and to send the request to the database; the comparison itself takes place in the database. As soon as the comparison is finished, the controller receives a list with users that have to be notified. If this list is empty, the process stops at this stage. Else, an event is created and the controller forwards the event together with the list to the messenger, which finally sends a notification to the respective users.

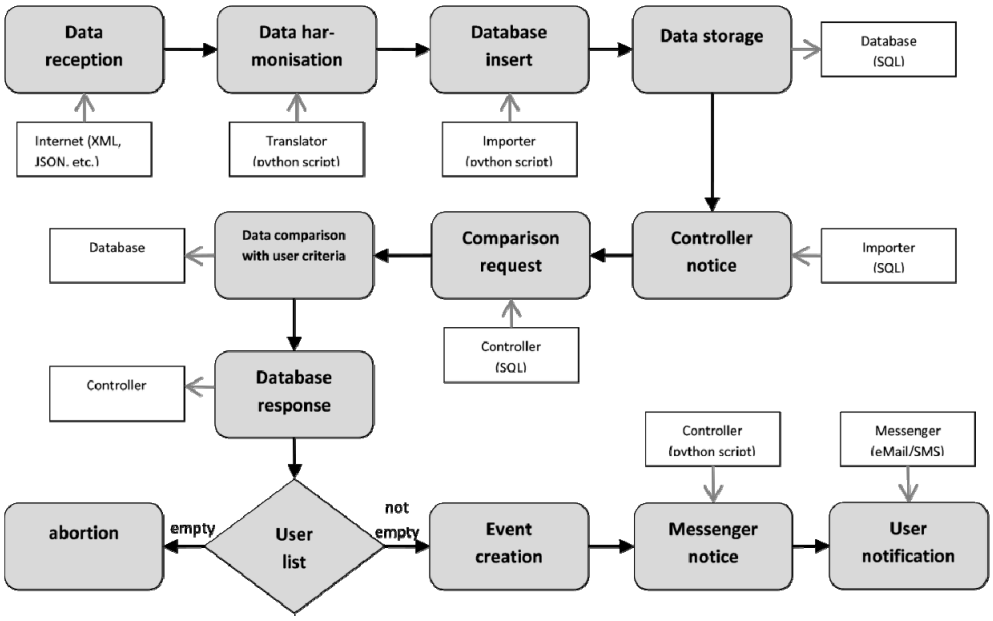

Fig. 4: Process chart of a typical event handling procedure (authors' design)

A schematic overview on the exemplary implementation of the filtering and notification service is provided in figure 5. As shown, the framework is divided into a client and a server part. While the client-side consists of the UI, the server-side encompasses several components, which are explained in the following. The central element of the server part is the controller. It is connected to all other modules, and is therefore able to trigger the import of new data, the comparison request, and the notification of the users. There are three different fashions how new data may be imported: a specific translator listens to incoming data from push services, a time dependent import of data from pull-services can be triggered by the operating system (e.g. cronjobs), or new data is pulled, based on an explicit user request via the Web interface.

The Web application backend, including a mapserver, represents the server-side of the UI, allowing for the interaction with current maps, as well as the manipulation of the areas of interest and user data. The client-side of the Web application is developed in JavaScript and uses OpenLayers 3, jQuery and jQuery-UI.

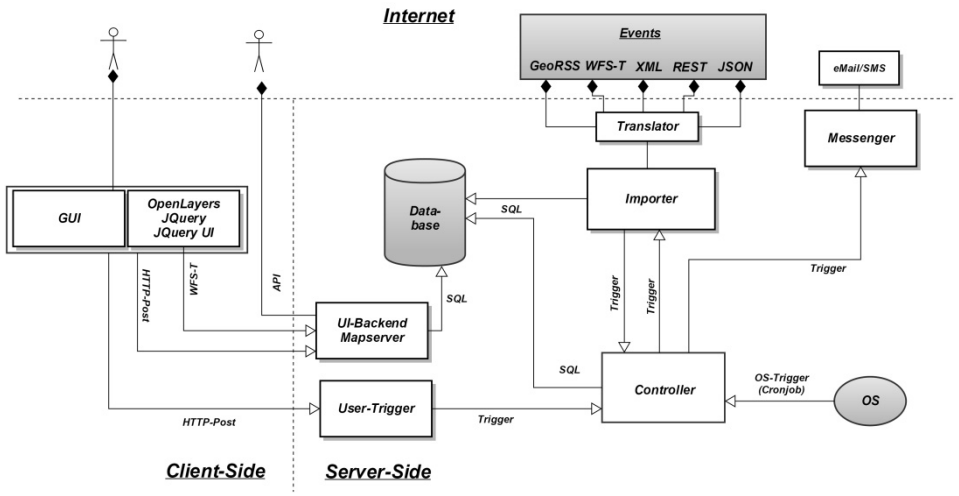

Fig. 5: Architecture of the notification system (authors' design)

Turning to the GUI, a map is the central element, representing current events of services that were generated through the framework. If the user is logged in he/she is able to subscribe to one or more services or resources and add the area of interest by drawing a polygon on the map. Once a polygon is drawn, the user is asked to define (non-geographic) parameters, which are utilised to filter received data. Parameters include time restrictions and other thresholds (e.g. to define a certain wind speed or the wave height). Also, the User Account button changes its label to the respective username, and enables general user settings (e.g. the channel of notification, etc.).

The database is in charge of storing the user data as well as the input data. Further, it facilitates the joining of relations, and filters the data according to the parameters defined by the user. Particular conditions encompass the following:

- A user is able to subscribe to multiple services.
- A user is able to define multiple areas for each service and for each area determine multiple non-geographic thresholds.
- A history of the source data should be kept.

A conceptual model of the database design is provided in figure 6. As shown, two sections exist, namely the user data and the service data. The main table of the user data is the **Subscriber** table. The join table **Subscriber_Service** table the user to subscribe to multiple services. For each subscription, multiple areas of interest are stored in the table **Subscriber_defined_Location** and their non-geographic parameters in the table **Subscriber_defined_Parameter**. The **Input** table stores the main information of the source data. Each occurrence is associated with a service and may have additional non-geographic parameters, which are stored in the table **Input_Parameter**. For performance reason, the table **Active_Input** stores currently active occurrences only. The table **Default_Parameters** contains the list of possible non-geographic parameters for a service.

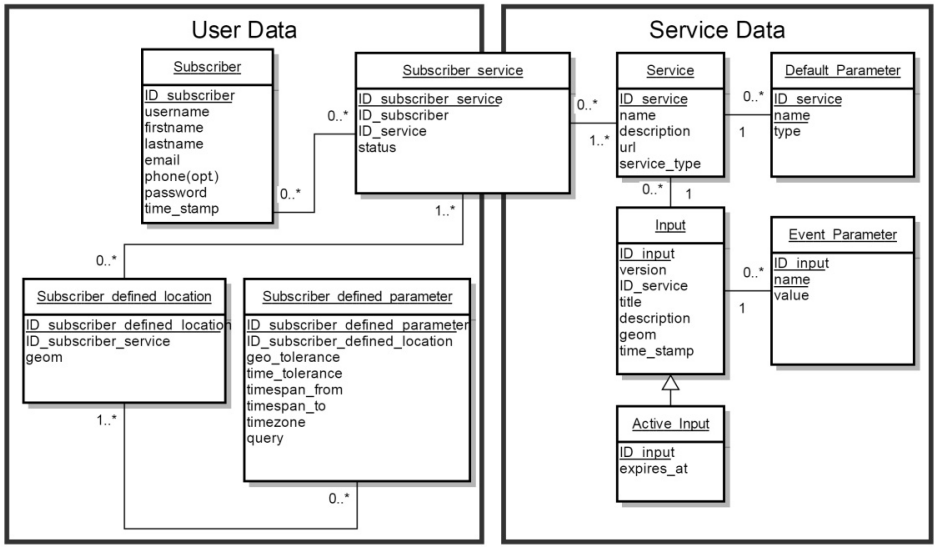

Fig. 6: Conceptual model of the database design (authors' design)

5 Discussion and Outlook

The conceptual framework at hand is meant to help users in dealing with the increasing amount of spatial data and data sources by transforming them into user-optimised, individual information. Four key components were identified to develop an architecture, which fulfills this requirement. Firstly, the user is able to control the notifications that should be received by defining various parameters using a GUI. This UI is dedicated to people without GIS experience. Secondly, a spatial database stores and combines user and service data. The usage of spatial and non-spatial indices, as well as proven data structures and algorithms help to provide fast response times without additional software components. Thirdly,

connectors to existing data sources collect and harmonise input data. Finally, an alerting component combines all components, and transmits notifications to the users. To facilitate an easy and cost-efficient integration of the framework into existing systems, open source products are utilised, and particular attention was paid to avoiding software dependencies during the design phase.

With respect to the future development of the framework, the integration of various other formats and services should be a major aim. A first challenge in this context could be the integration of streaming data. This could be achieved by using CEP approaches. However, this would mean to introduce a large dependency. Thus, in a first step it should be investigated how frequently the streaming of spatial data is used, and how large the potential target group would be. Besides, the integration of proprietary formats such as the Shapefile could increase the diversity of source data, and thus create further areas of application. In addition, the optimisation and enhancement of the UI for mobile devices could be an important step. For example, a person observing animals in the field may be interested in receiving notifications of particular events based on his/her current position. Another aspect would be the consolidation and intersection of data from different Web services and formats, facilitating the generation and provision of new information for the user. The architecture of the framework, as well as the database as a central component, can be customised with few changes. Last, improvements could be achieved by integrating fuzzy logic into the decision process. This could be applied to several parameters for both single and multiple events, and in sequence. The user would be notified if an overall status is satisfied, even if single thresholds are not met. As an example, this could include a geographic tolerance, transmitting events which happen nearby the defined area.

References

ECKERT, M. & BRY, F. (2009), Complex Event Processing (CEP). München.

GYLLSTROM, D., WU, E., CHAE, H., DIAO, Y., STAHLBERG, P. & ANDERSON, G. (2006), SASE: Complex event processing over streams.

OPEN GEOSPATIAL CONSORTIUM (2006), Draft OpenGIS® Web Notification Service Implementation Specification. OGC Document 06-095.

OHIO DEPARTMENT OF ADMINISTRATIVE SERVICES (2011), Ohio's Location Based Response System: How one set of highly accurate, shared mapping data is saving time, money and lives across the Buckeye State. Columbus, Ohio. http://gis3.oit.ohio.gov/LBRS/_downloads/docs/White%20Paper-LBRS_2011.pdf (30.01.2015).

ROBINS, D. (2010), Complex event processing. Second International Workshop on Education Technology and Computer Science. Wuhan.

SAMET, H., SANKARANARAYANAN, J., LIEBERMAN, M., ADELFIO, M., FRUIN, B., LOTKOWSKI, J., PANOZZO, D., SPERLING, J. & TEITLER, B. (2014), Reading News with Maps by Exploiting Spatial Synonyms. Communications of the ACM, 57 (10), 64-77.

SHASHA, D. & BONNET, P. (2003, Database Tuning: Principles, Experiments, and Troubleshooting Techniques. Morgan Kaufmann Publishers, Burlington.

WILLIAMS, R. (1987), Selling a geographical information system to government policy makers. Annual Conference of the Urban and Regional Information Systems Association.