A Framework for Web and Mobile Volunteered Geographic Information Applications

Clodoveu A. Davis Jr., Hugo de Souza Vellozo, Michele Brito Pinheiro

Depto. de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)
Av. Pres. Antônio Carlos, 6627 – 31.230-010 – Belo Horizonte – MG – Brasil

<clodoveu,vellozo,mibrito>dcc.ufmg.br

Abstract. *This paper proposes a framework to be used in the creation of various volunteered geographic information applications, incorporating both Web-based tools and mobile applications (apps). The framework includes the elements required to customize the information collection, while using a unified structure and interface across Web and mobile platforms. We demonstrate the use of the proposed framework in an application, named Strepitus, which gathers information on noise sources, so that users can report abusive sources of noise disturbance. The framework's elements and the components of Strepitus are presented in tandem, so that the design decisions are made clear. We observe that the framework allows for a quick development of new VGI applications, and this is important for capturing data on phenomena of current interest for the users, thus helping to maximize the number of volunteered contributions.*

1. Introduction

The quantity and variety of geographic data available on the Web for the use of the common citizen have been growing quickly. Since the introduction of Google Earth, in 2004, and of Google Maps, in 2006, there has been an increasing interest on tools that allow people to locate points of their interest and on applications that offer location-based services, such as finding urban addresses and selecting routes for vehicles. This kind of widespread interest in online maps and applications has led to an increasing demand for detail, especially in urban areas, and for currentness. People are quick to criticize a map for being outdated, not taking into consideration the cost and effort that would be required to keep huge volumes of data up to date. On the other hand, since the Web 2.0 proposal, several examples of situations in which users are allowed to contribute with the updating effort, or to supply more information, have come up, in a phenomenon usually called *Volunteered Geographic Information* (VGI) (Goodchild 2007; Goodchild 2007).

There are some online systems and Web sites that specialize in gathering user-generated spatial content. Most focus a specific theme, but overall the structure and working mechanisms of these tools is quite similar. Nevertheless, a new implementation is required for each different theme, so that parameters such as the geographic representation alternative (point, line, polygon), attributes to be collected, user identification, information validation and user feedback are adapted to the specificities of the theme and of the contributing crowd. The variety of these parameters makes it difficult to implement reusable solutions.

As a contribution to solve this problem, we propose a framework to be used in the creation of various volunteered geographic information applications, incorporating both Web-based tools and mobile applications (apps). The framework includes the elements required to customize the information collection, while allowing for a unified structure and interface across Web and mobile platforms. We demonstrate the use of the proposed framework in an application, named *Strepitus*, which gathers information on noise sources, through which users can report abusive sources of noise disturbance.

This article is organized as follows. Section 2 presents concepts and related work. Section 3 presents the framework's general architecture and components. Section 4 describes the application of the framework into the creation of Strepitus. Section 5 shows an analysis of the current version of the framework and its potential uses. Finally, Section 6 presents our final remarks and discusses further work.

2. Related Work

The mechanisms that lead people to spend time and effort to produce for free something that may or may not be useful to others are yet unclear (Budhathoki 2007; Goodchild 2007). However, the basic rule of the *wiki* mechanisms (originally from open source software development) seems to be valid for a wide range of situations: "if there are enough eyes, all *bugs* are shallow" (Raymond 1999). This means that if a dataset that is freely available on the Web is interesting for a large group of people, chances are that a reasonable share of these people would be willing to pay back the services they use through a disposition towards helping to improve the service for their peers. Such a phenomenon has been studied in Communications, where it is known as "collective action" (Bimber, Flanagin et al. 2005; Frew 2007), and directed towards public information stores, called *information commons* (Onsrud, Câmara et al. 2004).

Regarding geographic information, a similar phenomenon has been taking place in the last few years, under the generic denomination of Volunteered Geographic Information (VGI) (Goodchild 2007), but also known as spatial crowdsourcing, public participatory GIS, and other names. Starting with online maps and georeferenced image sets, such as Google Maps, Bing Maps, Yahoo Maps and others, several Web sites have been created counting on the perspective collaboration of users in the creation and/or maintenance of a niche geographic dataset. Through such mechanisms, any user, recognizing some aspect of the surrounding reality, can create an annotation, a comment, provide a place name, or connect a location to another source of complementary information. The prime example is perhaps the OpenStreetMap[1] project, with which users can contribute by editing street networks and providing an array of additional urban information. The resulting maps can be exported and used, costlessly, under an Open Database license.

Some studies (Haklay, Singleton et al. 2008; Keen 2008; Parker, May et al. 2012) show that arguments on VGI data being inferior to Professional Geographic Information (PGI) are invalid. User satisfaction derives from knowing when and how to use PGI, VGI or both. Volunteered information can be incomplete or inaccurate, but there are situations in which the only sources of detailed or thematically-relevant

[1] http://www.openstreetmap.org

[2]http://www.em.com.br/app/noticia/gerais/2012/06/07/interna_gerais,298842/denuncias-de-excesso-

information are volunteered. Therefore, VGI related projects still face numerous challenges: (1) how to effectively disseminate the data collection effort and how to motivate the largest number of people to contribute (Maué 2007), (2) how to maintain the contributors interested and active in the continuation of the effort (Coleman, Georgiadou et al. 2009; Soares and Santos 2011), (3) how to provide feedback to the contributors, generating personal or community benefits, and (4) how to validate and establish trust in collaboratively generated data (Frew 2007; Flanagin and Metzger 2008).

Our group has previously presented a generic platform for collecting and filtering volunteered geographic data (Silva and Davis Jr 2008). That initial effort was hindered by the use of proprietary software and lack of flexibility in the underlying structure. Sheppard (2012) presented a similar framework, with a high degree of reusability for VGI applications, using open source components and open standards, such as HTML5. One of the strong points of Sheppard's proposal is the use of generic code, which can run both in Web and mobile platforms. Unlike Sheppard's framework, we propose, within our framework, a set of structures that are designed to take advantage of specific hardware features in each of these environments. Besides, we use the Model-View-Controller design pattern (Reenskaug and Coplien 2009), and exercise a preference towards OGC-capable components. Therefore, our new approach, presented in the next sections, is a completely reengineered version of the previous work, adding a new architecture and new features, such as the integration of mobile platforms through the use of specific apps, while seeking a broad scope of applications on top of a simple and flexible structure.

3. Framework Architecture and Design

The proposed framework's architecture has been designed with the objective of encapsulating a basic structure for VGI applications into blocks that can be easily extended and reused in new projects. Basically, a new VGI application can be defined after some decisions are made, regarding (1) the geographic representation alternative for the collected data, (2) the required attributes, (3) the user management policies (login, history of contributions, reputation), (4) the validation policies, (5) interface elements for data inclusion and querying, and (6) user feedback, in terms of the display of the contributions, analysis of accumulated contributions, and others.

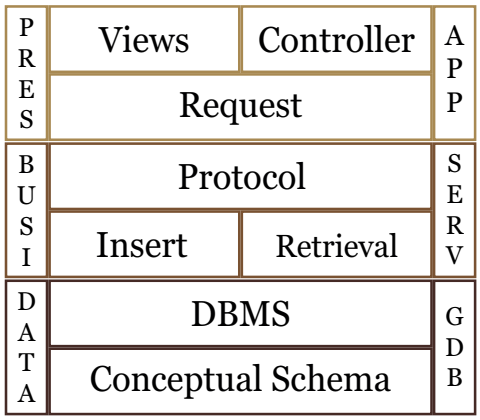

Figure 1 – General framework architecture

Based in our previous work, the framework's components are organized into three layers: Presentation, Business and Data (Figure 1). Details on the internal data components of each layer are depicted in Figure 2. The Presentation layer contains the modules that are responsible for data collection itself, implemented on multiple environments, such as Web and mobile. Currently, we have components for iOS and Android app creation. In the Business layer we concentrate services that perform the communication between the Data layer and the Presentation layer. The Data layer contains the geographic database that is responsible for storing the collected data.

The Data layer is defined within a spatial DBMS, after a conceptual modeling for the VGI application. The conceptual schema for the resulting database has been standardized for the framework, using a generic schema with three groups of objects: *users*, *contributions* and *contribution assessment*. In the *users* group, classes called User, LogUser and UserType are included. The first manages user accounts, including login names and passwords. The second records user activity in the system, for further analysis and for defining user trust and reputation. The third supports user classification into groups that are meaningful for the application. Functions define configuration alternatives such as (1) modes of user identification to access the system or to contribute, (2) raising users to moderator status, (3) the policies for establishing the user's reputation in the system, among others.

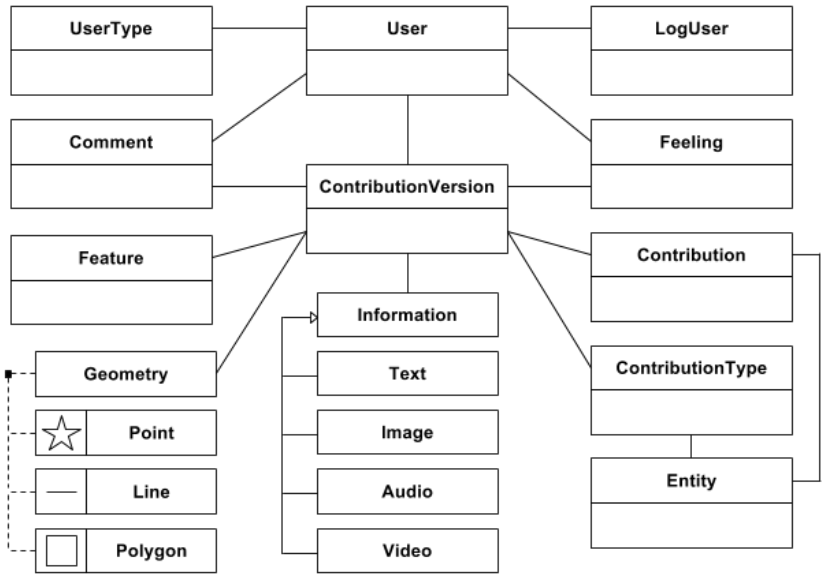

Figure 2 – Class diagram for the framework's generic database

In the *contributions* group, five classes are defined, namely *Contribution*, *ContributionVersion*, *ContributionType*, *Entity* and *Feature*. Each contribution can be edited to include further textual information, or associated to image, audio or video data. Contributions can be represented using points, lines or polygons. All versions of a contribution are stored, so that unmoderated contributions can be distinguished from ones that have undergone some revision. The *Entity* class allows for the definition of multithematic contributions, i.e., applications in which the user can provide data on

more than one theme. Furthermore, multiple types of contributions can exist for a given entity, so that variations in the definition of entities can be accommodated.

The third group of classes encompasses the ones that contain data on the assessment of user contributions. There are many validation strategies for VGI data, including peer validation and moderation. The *Feeling* class records peer opinions (rebuttals or confirmations, for instance). Additional textual data are recorded by the Comments class. Naturally, the validation mechanism and policies for filtering out inadequate contributions vary according to the data collection effort, and our intention with the framework is to provide a range of options for the creation of new applications.

Services in the Business layer are divided info three main components: *Protocol*, *Insertion and Retrieval Services*, and *Access Drivers*. The first component defines how the service will be available to the applications, i.e., the network protocol to be used in the exchange of messages, and their format. This is important for the separation of the Presentation layer, so that it can be implemented in multiple platforms. The *Insertion and Retrieval* services can be customized in order to accommodate the requirements of the applications and to respect constraint imposed by the database management system. The *Access Drivers* vary according to the DBMS used in the Data layer and to the computational environment for the implementation of the services.

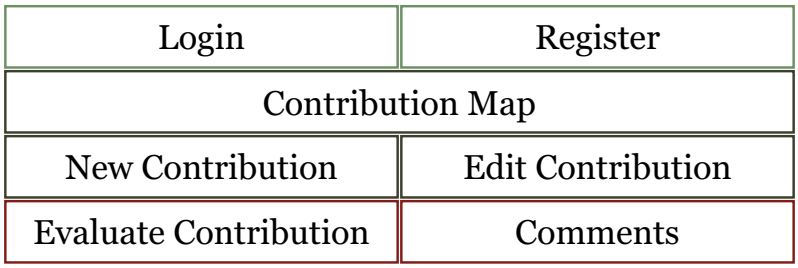

Figure 3 – Application view details

Finally, the Presentation layer implements the applications' user interface, which can be described using a Model-View-Controller design pattern. According to the pattern, the applications are defined as a set of *Views* that supply information to the *Controllers*. These, in turn, process the data supplied by the users along with data received by the Request module (our *Model* component, details below), which works as the input and output of application requests for the services. In the Views set, three subgroups have been defined, related to the groups of classes mentioned in the Data layer. Views for login and user registry supply and receive information from the first subgroup of data classes, implementing user creation and application access code. Views for the contributions in the map, attribute input and editing are also defined, and related to the second group of data classes. Finally, views for assessing contributions and comments input are coordinated with the third group of data classes.

Controller components are used to pre-process information. Such components can use, along with user-supplied data, information from the environment in which they have been developed. This is very common in mobile applications, in which the hardware is responsible for supplying data on the device's geographic location, using GPS or A-GPS hardware. Likewise, Web applications can return a geographic

coordinate derived from a mouse click, by inverting the projection transformation used to display a map or georeferenced image.

Our Model component is called Request, because its primary function is to send and receive information to the services. This module is able to handle one or more services, depending on the complexity of the application and on the resources that are available in the implementation environment.

4. Strepitus: a VGI application for noise mapping

4.1 Motivation

Living in large cities is getting more and more stressful, due to daily modern life problems. Activities that generate environmental impact are sometimes beyond the control of governmental authorities. The proximity of such impact-generating activities and housing districts produces a rising number of conflicts, often generating discussions that end up in police reports or even law suits.

In this context, disturbances caused by excessive noise levels are contrast with the need for a relative silence in resting periods. The Brazilian standard NBR-10151 establishes a limit of 60 dB for nightly noise in predominantly industrial areas, and 50 dB in residential areas. These limits are often exceeded in large cities. In Belo Horizonte, the number of complaints to governmental authorities about excessive noise has risen by 30% in 2012, relatively to 2011[2]. In São Paulo, a survey determined that about 20% of calls to the police over weekends are noise-related[3]. In Rio de Janeiro, noise pollution operations take place in the early hours of the day from Friday to Sunday, leading to police arrests[4]. However, the environmental laws and the action of the authorities are not powerful enough to keep the problems from happening, sometimes because of the difficulty to produce evidence on a problem that is geographically dispersed, does not occur all the time, and is more intense in out-of-office hours.

Inspired by the intensity and pervasiveness of noise complaints, we used the framework described in the last section to create a VGI application (named *Strepitus*) that is able to receive input from citizens on excessive noise levels. Strepitus was designed to let users indicate the location affected by a noise source, adding data on the sound intensity as an attribute. Since regulations have different limits for indoor and outdoor environments, the user also indicates whether the noise level has been recorded indoors or outdoors.

In mobile platforms, Strepitus is able to estimate the noise level using the device's microphone. Users are required to log in to the system, creating an account if necessary. After logging in, they can record a contribution. In the Web application, the user must indicate her location over a base map, and then fill out the attributes (noise level, indoors/outdoors) on the screen. In the mobile apps the position is obtained from

[2]http://www.em.com.br/app/noticia/gerais/2012/06/07/interna_gerais,298842/denuncias-de-excesso-debarulho-sobem-38-em-bh.shtml

[3] http://www.estadao.com.br/noticias/impresso,policiais-vao-fiscalizar-lei-do-silencio-,1010397,0.htm

[4]http://g1.globo.com/ma/maranhao/noticia/2012/05/em-cinco-meses-combate-poluicao-sonora-japrendeu-69-em-sao-luis.html

the device's operating system location functions, and the noise level is determined from the microphone interface.

The next subsections present the mapping from the framework to an actual VGI tool, along with elements from its operation.

4.2 Architecture

Along the lines of the proposed framework, Strepitus implements the three previously mentioned layers: Presentation, Business and Data. In the Data layer, a central database server has been configured to receive and maintain all necessary data. The Business layer includes a Web service provider and a metadata catalog server, which offer data access that goes beyond the need of the VGI application itself, organizing access to the underlying data tables. In the Presentation layer, two mobile applications have been developed, for the iOS and Android operating systems, along with a Web application.

Figure 4 – Strepitus architecture

The generalized data schema described in Section 3 has been adapted so that the User, LogUser and ContributionType classes are defined for Strepitus. The ContributionType class holds information on noise sources, which correspond to classes of contributions in the framework. This is actually a table that holds typical sound intensity for common sources that serve as references to the users (Table 1).

Table 1. Noise sources and corresponding sound intensities

Noise source (Contribution type)	Typical intensity (dB)
Library	40
Conversation at home	50
Air conditioner	60
Dog bark	65
Vacuum cleaner	70
Classroom	75
Large volume of street traffic	80
Construction	100
Live show/concert	115
Sporting event	120
Party	120
Pneumatic hammer	130
Jet engine	135

For Strepitus, we merged the ContributionVersion, Contribution and Feature classes, storing all the volunteered data in the same structure. This alternative applies to situations in which editing is not allowed, and the attribute set is simple. In Strepitus,

values are not editable, since the noise level is typically captured by the hardware and events have a short time span.

In the Business layer, the framework was used to support the creation of a Web service running under PHP. This service runs from the same server that hosts the database server, but it can be directed to a separate machine, depending on the expected volume of user requests. According to the framework's architecture, HTTP is used as the message transmission protocol, and messages are encoded using JSON. The service offers mediation in the insertion of new users, contributions and application accesses, as well as for data recovery on the existence of users and on contributions that must be presented over a base map. For the access drivers, we used PHP libraries that support connections to PostGreSQL (PostGIS). Geoserver is also included in the Business layer, serving as a tool that provides data to be displayed along with application maps. As an application server, Geoserver supports connections to several different data sources and is able to publish them as OGC Web services (Percivall 2003). It also provides a metadata catalog. Using this component, additional layers can be presented over the base map, combining data from various distinct sources.

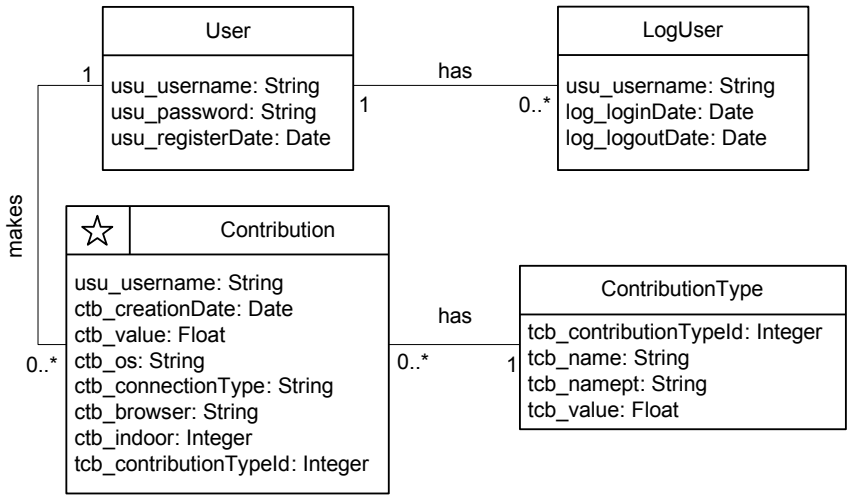

Figure 5 – Strepitus' database class diagram

In the Presentation layer, along with a Web application, mobile apps have been developed (Figures 1 and 2). Mobile development environments generally suggest the use of the MVC design pattern, so that the user interface does not have to be interrupted during data requests, and can therefore maintain processes dedicated to view updating. For the mobile apps, iOS and Android platforms were selected, due to their current popularity. The same set of user interface functions was developed for each platform, including login and registering screens, contributions map, contribution data collection, configurations, and help information. Special Controller modules have also been developed for each mobile platform, in order to get position data and noise intensity from the corresponding hardware components of the mobile computers. Finally, there are also framework components to send user requests to the database server, using the previously defined protocol, mentioned earlier.

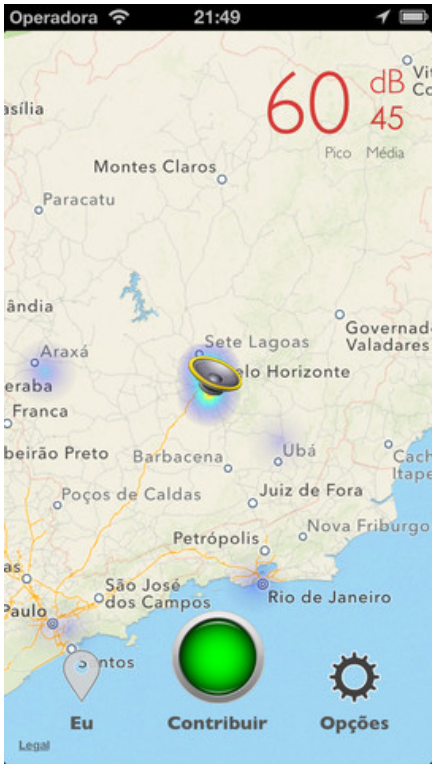

Figure 1 –iOS app

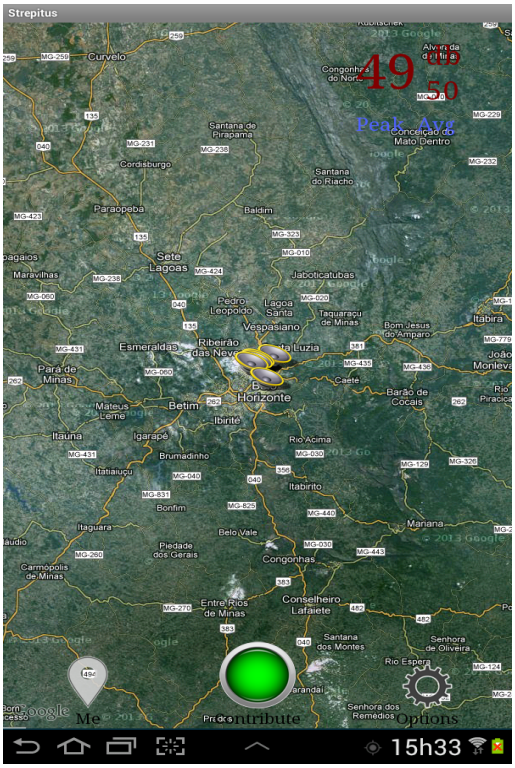

Figure 2 –Android app

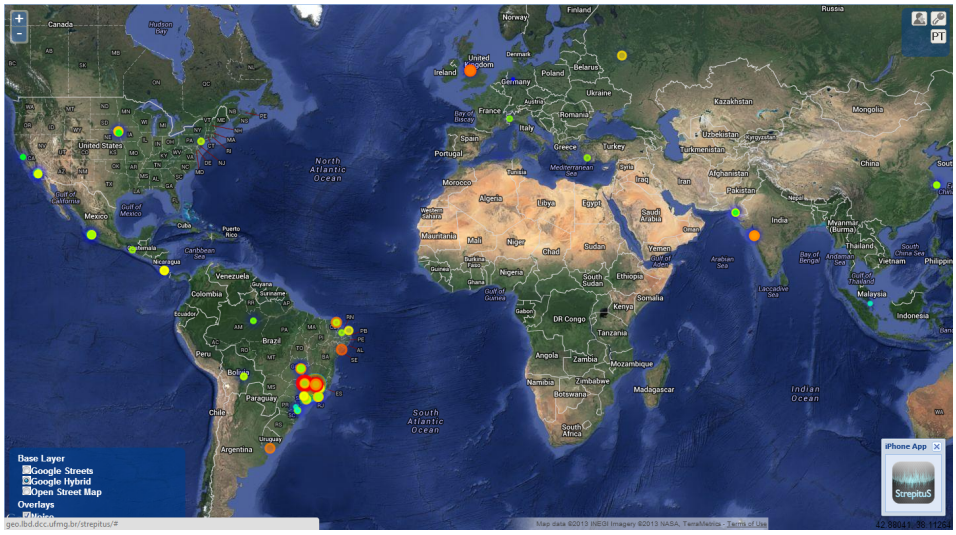

Figure 3 – Web application

The Web application (Figure 3) was developed using PHP and JavaScript, also using libraries such as OpenLayers, ExtJS and GeoExt. The user interface functions

developed for the mobile platforms were adapted to smaller windows, instead of using up all the screen space. Since the Web application runs in machines that typically are not equipped to measure noise, the user is provided with a reference scale of noise sources, so that the most approximate perceived noise level can be selected for the contribution (Table 1). The main Controller module used in the Web application is the one in which the user indicates the geographic position for the contribution. Among the Model components, the one responsible for requesting base map layers uses a communications protocol dedicated to OGC Web services.

5. Conclusions and Future Work

This paper presented a framework for the creation of VGI applications, capable of coordinating contributions from Web and mobile apps. Our framework includes all the components required to implement various VGI applications, aiming to reduce effort of preparing and publishing new VGI themes. In fact, our intention is to be able to quickly design and launch new VGI applications directed at current events, so that more people can be motivated to contribute.

Some components are still needed for the framework. In the near future, we intend to incorporate a user identification and login function that allows for the use of social network IDs, as in the case of Facebook and Twitter. Some integration with these social media is also sought, so that a contributor can simultaneously provide data and invite her friends to participate. We are also investing in a wider array of validation functions, in order to allow for user-based confirmation of peer-posted contributions. There is also space for additional visualization functions, to be used to provide visual feedback to users as to the universe of contributions and their geographic distribution.

As to Strepitus, the application is currently online, and the iOS and Android apps can be downloaded from the respective app stores. This application allowed us not only to test the ideas that led to the framework, but also to learn about the publication process for today's mobile apps. Since its publication, Strepitus has recorded contributions from all over the world, although only a small number of them, since no publicity campaign was made for its use. As previously mentioned, user motivation is part of the most relevant challenges for VGI research, and it is probably much easier to gain popular support to themes that are currently under intensive coverage by the media – and for that reason we intend to be able to generate VGI applications very quickly from the proposed framework.

Acknowledgments

The authors acknowledge the support by CNPq (560027/2010-9, 308678/2012-5) and FAPEMIG (CEX-PPM-00518/13), Brazilian agencies in charge of fostering research and development.

References

Bimber, B., A. J. Flanagin, et al. (2005). "Reconceptualizing collective action in the contemporary media environment." Communication Theory **15**(4): 365-388.

Budhathoki, N. R. (2007). Reconceptualization of user is essential to expand the voluntary creation and supply of spatial information. Workshop on Volunteered Geographic Information, Santa Barbara, California, USA.

Coleman, D. J., Y. Georgiadou, et al. (2009). "Volunteered geographic information: the nature and motivation of produsers." International Journal of Spatial Data Infrastructures Research **4**: 332-358.

Flanagin, A. J. and M. J. Metzger (2008). "The credibility of volunteered geographic information." GeoJournal **72**(3): 137-148.

Frew, J. (2007). Provenance and volunteered geographic information. Workshop on Volunteered Geographic Information, Santa Barbara, California, USA.

Goodchild, M. F. (2007). "Citizens as sensors: the world of volunteered geography." GeoJournal **69**(4): 211-221.

Goodchild, M. F. (2007). "Citizens as Voluntary Sensors: Spatial Data Infrastructure in the World of Web 2.0." International Journal of Spatial Data Infrastructures Research **2**: 24-32.

Haklay, M., A. Singleton, et al. (2008). "Web mapping 2.0: The neogeography of the GeoWeb." Geography Compass **2**(6): 2011-2039.

Keen, A. (2008). The Cult of the Amateur: How blogs, MySpace, YouTube, and the rest of today's user-generated media are destroying our economy, our culture, and our values, Random House Digital.

Maué, P. (2007). Reputation as a tool to ensure validity of VGI. Workshop on Volunteered Geographic Information, Santa Barbara, California, USA.

Onsrud, H., G. Câmara, et al. (2004). Public Commons of Geographic Data: research and development challenges. GIScience 2004 (LNCS 3234). M. J. Egenhofer, C. Freksa and H. J. Miller. Berlin/Heidelberg, Springer-Verlag**:** 223-238.

Parker, C. J., A. May, et al. (2012). "The role of VGI and PGI in supporting outdoor activities." Applied Ergonomics(In press).

Percivall, G. (2003). OpenGIS Reference Model, Open Geospatial Consortium, Inc.

Raymond, E. (1999). "The Cathedral and the Bazaar." from Available at http://www.tuxedo.org/~esr/writings/cathedral-bazaar/.

Reenskaug, T. and J. O. Coplien (2009) "The DCI Architecture: A New Vision of Object-Oriented Programming."

Sheppard, S. A. (2012). wq: A Modular Framework for Collecting, Storing, and Utilizing Experiential VGI. 1st ACM SIGSPATIAL International Workshop on Crowdsourced and Volunteered Geographic Information (GeoCrowd'12). Redondo Beach, CA**:** 62-69.

Silva, J. C. T. and C. A. Davis Jr (2008). Um framework para coleta e filtragem de dados geográficos fornecidos voluntariamente. X Brazilian Symposium on GeoInformatics (GeoInfo 2008), Rio de Janeiro (RJ), Sociedade Brasileira de Computação.

Soares, M. D. and R. Santos (2011). Ciência cidadã: o envolvimento popular em atividades científicas. Ciência Hoje. **47:** 38-43.