Volunteered Geographic Services: Developing a Linked Data Driven Location-based Service

Alexander Savelyev
Pennsylvania State University, USA
azs5362@psu.edu

Krzysztof Janowicz
University of California, Santa Barbara, USA
jano@geog.ucsb.edu

Jim Thatcher
Clark University, USA
jethatcher@gmail.com

Sen Xu
Pennsylvania State University, USA
senxu@psu.edu

Christoph Mülligann
University of Münster, Germany
cmuelligann@uni-muenster.de

Wei Luo
Pennsylvania State University, USA
wul132@psu.edu

ABSTRACT

The term Volunteered Geographic Information (VGI) describes various layperson-based, geo-collaboration projects to collect, maintain, and visualize information. VGI has been successfully utilized in scenarios such as emergency response and is also increasingly integrated into commercial products. Based on an analysis of existing projects and research, we propose to extend the idea of VGI by introducing Volunteered Geographic Services (VGS). Instead of contributing information, volunteers can request or offer micro-services to their local community. We provide a flexible server framework that handles service requests and offers. We also implement a smartphone application developed using Google's Android platform. The server and mobile client are realized following the Linked Data paradigm and using Semantic Web technologies. In this paper, we discuss the idea behind VGS, motivate it using two scenarios, and explain the technical realization.

Keywords

Location-based Services, Volunteered Geographic Information, Linked Data, Mobile Computing

1. MOTIVATION

The term Volunteered Geographic Information (VGI) was coined by Goodchild [5] to describe the multitude of layperson contributions to the field of geographic information science or geography in general. The most well-known VGI projects are OpenStreetMap[1], WikiMapia[2], and Ushahidi[3]. Presently, VGI is predominantly centered around collaborative mapping and visualization. OpenStreetMap, for example, aims at developing a freely accessible and editable map of the earth, while Ushahidi maps cases of political repression and has shifted to other kinds of emergence reports since the Haiti earthquake [10]. Despite the diversity in their contents and user communities, all VGI projects share the vision of collaborative data creation and maintenance, and can be regarded as special geospatial cases of user-generated content [5]. The provenance and quality of geospatial collaborative data have been a recent topic of research: Zook et al. [17] studied the role of VGI during the Haiti earthquake, while Goodchild and Glennon [6] investigated the wildfires in Santa Barbara, California. Both reports highlight the specific role of crowdsourcing in VGI, i.e., the assumption that data collected by several contributors is more likely to reflect the situation on the ground than information provided by a single, authoritative observer.

Based on existing research on VGI, Volunteered Geographic Services (VGS) may take geo-collaboration to the next level. In contrast to providing data, the motivation behind VGS is to develop an infrastructure to contribute and exchange micro-services. Users may announce service offerings or publish service requests. The type of such human services is not limited by the framework and may range from mere courtesies to serious emergency assistance. The former case, may involve a user asking for the current driving-related weather conditions in a location, something fulfilled by another user looking out of her window. In the latter, a user could request help in removing a fallen tree from the street or even offering transport during a natural disaster such as a flood or a storm. Automatic matchmaking approaches for such micro-services were recently proposed by Raubal and Winter [12]. The VGS system allows users to see which requests and offers have been published and how they are linked to each other, e.g., whether a particular request has already been answered by an offering. Essentially, Volunteered Geographic Services are a human-centered kind of Location-Based Services (LBS), one that allows offering and requesting through the use of mobile devices such as Android-based smartphones.

The remainder of this paper is structured as follows: two hypothetical scenarios are described as a means of clarifying the role of VGS and to place it in relation to other projects

[1]www.openstreetmap.org
[2]www.wikimapia.org
[3]www.ushahidi.com

such as SeeClickFix.[4] Next, the Linked Data model and the VGS Service Bus are introduced. Afterwards, the mobile client is presented. Finally, the paper concludes with a summary, lessons learned, and an outlook to future research.

2. SCENARIOS FOR VGS

The concept of Volunteered Geographic Services, like VGI, is not restricted to a specific application area; however, the following two scenarios highlight both its relevance and added functionality when compared to traditional VGI.

The first scenario describes a potential everyday use of VGS and is the running example for section 4.2. Snow shoveling is a typical early morning activity during winters in Central Pennsylvania. In case of unexpected snow accumulation, residents must schedule an additional one or two hours for shoveling. While some residents possess snowplows, others do not. Using the VGS system, a resident in a hurry could send out a service request for use of a snowplow. Nearby VGS users can answer the request and offer aid. While this case begins with a service request, the VGS system also allows users to create service offers. In this situation, a resident with a snowplow could create an offer and wait for other users to accept. The VGS Android application described in section 4.2 displays requests and offers on a map, supports spatial and temporal filters, and also shows whether a responder has already committed to offer support.

The second scenario depicts VGS utility in emergency management. According to the Report of the US National Research Council: *geospatial data and tools should be an essential part of all aspects of emergency management from planning for future events, through response and recovery, to mitigation of future events. Yet they are rarely recognized as such, because society consistently fails to invest sufficiently in preparing for future events, however inevitable they may be* [9, p.2]. The literature describes several successful applications of Volunteered Geographic Information in emergency scenarios such as the Haitian Earthquake, wildfires in California, and the Japanese earthquake [17, 6, 11]. In these cases, VGI was used to acquire massive, up-to-date data from the dynamic and complex situation on the ground. For instance, the OpenStreetMap community has re-mapped the transportation infrastructure after the earthquake in Haiti.

While this kind of data is crucial to the coordination of emergency response teams, it does not solve the support bottleneck: while volunteers collect data, task-supporting is coordinated only by sanctioned, authoritative parties. VGS offers a potential solution to this bottleneck, by allowing micro-services to be coordinated and performed by volunteers. VGS moves the idea of VGI one step further as not only authorities, but also the general public may act as emergency responders. Any citizen using a smartphone with GPS technology can provide aid on a peer-to-peer basis. For VGS to work in practice, the essential requirement component is a data server that allows users to submit help requests with location information and that affords potential responders the ability to search and browse help requests in their geospatial vicinity. In other words, and with respect to emergency response, geo-collaboration in VGS does not focus on providing data but on the collaborative coordination of action on the ground. Potential examples include users announcing the presence of functioning transportation, coordinating sandbag placement, or aiding in the removing of fallen trees.

[4]http://www.seeclickfix.com/

3. RELATED WORK

Goodchild [6] credited the rise of VGI to technological development, which allows average citizens to determine position accurately and gain map-making abilities. Accompanied by the rise of user-generated content, volunteered geographic data ranges from innocuous information such as snow quality on ski hills to maps of criminal violations [13]. While individuals may remain invisible to GPS [13], using VGI to address emergency response issues is increasingly common [17, 11]. VGI-based platforms such as Ushahidi rely upon citizens volunteering geographic information in order to aid workers in emergency response situations. VGI and VGS are closely related to crowdsourcing [7], which denotes the idea of using open calls to an undefined group of people to complete a (usually large-scale) task. Crowdsourcing has now been widely and successfully applied in business models under the assumption that the open call can draw a crowd that is best fitted for the task [8]. The success of VGS relies upon a self-established user community in which the citizens are not only willing to volunteer information but also to actively search and respond to requests within their means of action. Examples of such behavior include Airbnb (http://www.airbnb.com/), a social networking service taylored towards spur-of-the-moment rentals, and Avego (http://www.avego.com/), a transportation management platform with on-demand ride-sharing capability. However, actions may be as simple as looking out of the window while remaining useful to other users.

SeeClickFix is an emerging Web-based service which shares similar ideas with VGS. SeeClickFix allows users to report on non-emergency issues, such as broken windows or idling vehicles on street. It also allows government officials to monitor for important health and safety issues, such as cracked sidewalks or street crime. At the present, there is no existing framework for VGS that eases the implementation of services such as SeeClickFix. VGS applications can be as manifold as VGI, hence it is necessary to set a common ground for developers that aim at addressing different user communities. This can be solved through a generic request-offer messaging bus based on a Linked Data model [2]. All requests, offers, users, and their related elements, such as timestamps and locations, are openly available and can be combined with other sources. For instance, and in contrast to SeeClickFix, this framework can link requests to event data provided by sources like data.gov[5] or any other Linked Data source. The Android application presented in section 4.2 is just one tool running on top of the VGS infrastructure.

4. LINKED DATA MODEL

A service request asking for assistance may be answered by an offer, but also attached to another request to express that users share the same service needs. In the snowplowing case, the secondary request may indicate that another car is stuck in a nearby location. As these requests reinforce an existing request, they are not services in their own right. The more

[5]http://www.data.gov

generic term *action* is used when referring to all offers, requests, and their interlinkages. A network of interconnected requests and offers can also be used to establish trust in VGI [1], e.g., to filter out spam messages as well as providing a more complete picture of ongoing events. In this respect, VGS fits well within other work on the Semantic Sensor Web, e.g., to detect severe weather conditions [14]. In our VGS ontology, these interconnecting relationships are modeled by the *leadingAction* property and its sub-properties.

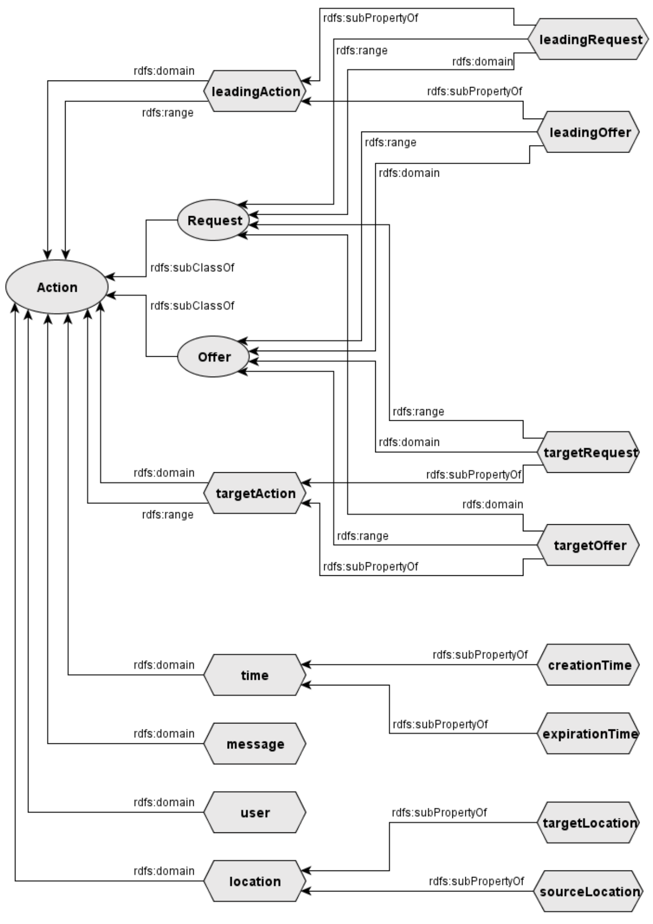


Figure 1: Classes (oval) and Properties (hexagonal) in the RDF schema of the VGS framework ontology.

In addition to interactions between requests and offers, an action may have different kinds of metadata: a user, his or her location, the location the action refers to, the time of creation and expiration, or the actual content of the message. This metadata is not modeled in the VGS framework's ontology as each specific VGS implementation may require different, application-level ontologies. Instead, the framework's ontology is able to incorporate external vocabularies such as FOAF[6]. The generic model allows basic server-side management of requests, offers, and associated information within the framework; see Figure 1. Functionality that goes beyond that, e.g., spatial indexing and queries, needs to be addressed by the actual implementation, which, for instance, might use geographical coordinates or place names combined with a gazetteer service to fill the *location* property. In other words, while application ontologies for specific VGS applications may introduce object properties and classes to model users, types of services, etc, the server side ontology does not restrict the range of the user, location, time, or message relations.

[6]Friend Of A Friend; see http://www.foaf-project.org/

The following listing shows an example of a request in N3 notation that connects to Geonames.org for place related information and FOAF for user profiles:

```
@prefix: <http://vgs.psumobile.org/core#> .
@prefix geo: <[...]w3.org[...]/geo/wgs84_pos#> .
@prefix rdf: <[...]w3.org/...]/22-rdf-syntax-ns#> .

[] a :Request ;
     :creationTime "2011-03-01T14:01:23"
      ^^<[...]/XMLSchema#dateTime> ;
     :leadingRequest <[...]/requests/request123/> ;
     :message  "Stuck in snow in front of
     Walker Bldg."^^<[...]/2001/XMLSchema#string> ,
      "3"^^<http://emergency.example.org/core#urgency> ;
      :sourceLocation
              [ geo:lat "40.794" ;
                geo:long "-77.866"
              ] ;
      :targetLocation <[...]geonames.org/7730361/about.rdf> ;
      :user   <[...]psu.edu/kuj13/foaf.rdf> .
```

In the example above, references are made to a number of distinct namespaces, including PSUMobile (empty prefix), Geonames.org (prefix "geo"), RDF Schema (prefix "rdf"), and FOAF. The location of this particular request is referenced through the property "sourceLocation" (Geonames.org namespace is referenced through its respectful prefix). User information is referenced through the property "user" and is specified with a full URI.

Following the Linked Data paradigm, each action, e.g., a service request, is identified using a URI and linked to other URIs representing service offers or support for this request. Data between the server and client applications is transferred via HTTP.

4.1 VGS Service Bus Implementation

This section outlines the current implementation of the VGS Service Bus, a core component of the server framework. Our use of the term "Service Bus" is different from that in general information domain. In the context of software architecture, enterprise service bus (ESB) describes the means by which independent software components (dubbed "services") communicate with each other. We use the term "services" in its literal meaning, as in a "helpful act" towards a person or a cause. "Service bus" is therefore seen as a medium for effective exchange of such services.

The bus has three main goals:

- Collect, store, and organize the actions, i.e., requests or offers, submitted by the VGS users.
- Allow users to perform spatial, temporal, and thematic queries over existing actions.
- Provide auxiliary services necessary for the maintenance of a large-scale, multi-user system.

In order to achieve these goals, a modular system design has been implemented. The main components of the VGS Service Bus are the Triploid API, VGS Web Service, Jena based Linked Data engine, and the PostgreSQL Database. Figure 2 illustrates these components along with their modes of interaction. The related transformations of the original *Action* class from the VGS clients to the database and back are also shown. Each component is described in greater detail below.

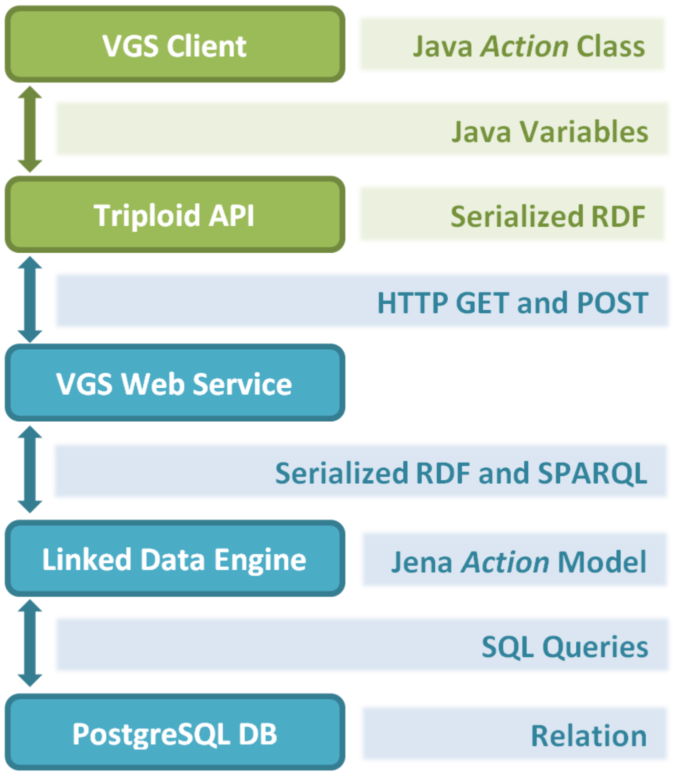


Figure 2: VGS Service Bus Implementation. Main components are shown as bold rectangles, their modes of interaction are shown as arrows, and the transformations of the original *Action* class are shown as faint rectangles. Components colored in green are deployed locally (on the client device), whereas the components colored in blue are deployed on the remote server.

From the perspective of the VGS Service Bus, the role of the VGS Client is to add new actions to the system and request sets of actions in the form of a query. This functionality, along with the end-user experience on the client side, is presented in more detail in this section. The Triploid API is a Java library that converts the original *Action* class, as generated by the client, into serialized RDF statements. The API is realized on top of Androjena 0.5 [7], a minimal port of the initial Jena framework to the Android platform. Staying within the realm of Semantic Web technologies and representation languages reduces the software engineering complexity and keeps the client (and server) more flexible in terms of future changes and extensions. In fact, moving the business logics from the source code in to the data is one of the core goals of the Semantic Web.

The VGS Web Service is the main gate through which the actions are exchanged between the client and the Linked Data engine. VGS Web Service abides by the principles of a RESTful service - it implements a uniform client-server interaction interface, is stateless, and uses HTTP methods (GET and POST) explicitly. It is implemented using Tomcat, i.e., based on Java Servlets. The VGS Web Service triggers functions in the Linked Data engine and handles the incoming actions and user-specified query parameters. Action URIs, user URIs, and other data designated for the client side are managed by the VGS Web Service as well. The Linked Data engine is realized using Jena RDF API and Jena SDB. It parses the incoming actions, forms SPARQL queries according to client specifications, retrieves individual actions from the database upon client request, and provides auxiliary services necessary for the maintenance of the system as a whole. Finally, a PostgreSQL database is used to store and organize the collection of actions generated by the VGS clients. Transparent storage of RDF data in the relational database is made possible by the Jena SDB API which provides the necessary functionality.

Whenever a new action is added to the system, the following events are triggered:

1. The action Java object is converted to RDF using our Triploid API.
2. The serialized RDF is sent to the VGS Web Service through a POST message and passed on to the Linked Data engine.
3. An action URI is generated and pushed to the PostgreSQL database along with other properties of the action received from the client side, e.g., the time-stamp.
4. The action URI is returned to the client as a confirmation of a successful data upload and to display the service offer or request on the client's map.

Whenever a set of query parameters is generated by the client, the following events are triggered:

1. Query parameters are sent to the VGS Web Service through a GET message and passed on to the Linked Data engine.
2. A SPARQL query is generated and Jena is accessing the underlying PostgreSQL Database.
3. Query results, i.e., RDF triples, are passed back to the client side.
4. A set of Java action objects is generated from the RDF triples using the Triploid API.

The resulting implementation of the VGS Service Bus is platform-independent, remissive of changes to the underlying Linked Data Model, and can be adopted to other VGS-based applications.

[7]http://code.google.com/p/androjena/

4.2 VGS Client Implementation

This section describes the implementation of the VGS client as a mobile application for the Google Android platform. The workflow of the client implementation is depicted in Figure 3.

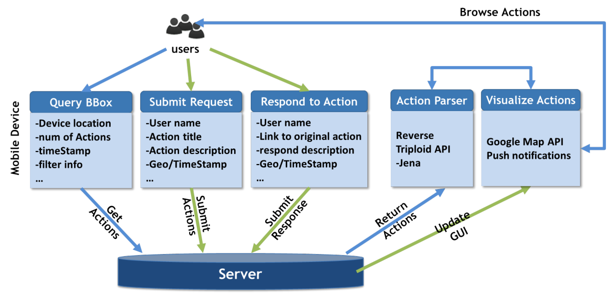


Figure 3: Workflow of the client to server communication.

The mobile client provides an interface for users to browse existing actions, submit new actions, and respond to existing actions. An action, in this context, is defined as a user initiated message, such as *requesting help* or *offering help*, and includes attributes such as *UserName*, *ActionTitle*, *ActionDescription*, *Location*, and *TimeStamp*. When the application is started, the default GUI is showing current actions in the user's vicinity. In Figure 3, the blue arrows depict the workflow for browsing existing actions. First, the device location is retrieved from the GPS component of the smartphone to perform spatial filtering using the map extent as a bounding box. The number of actions to retrieve is set to 10 as default, due to the display limitation on a mobile device; for an in depth discussion of this decision and the resulting consequences, see [15]. The *TimeStamp* for temporal filtering is set to the current time according to the VGS client. These parameters form a query that is passed on to the server and is then processed by the servlet to return the most 10 recent actions matching the spatial and temporal filters. The server's response is parsed by our Triploid API which generates action instances, i.e., Java objects. These actions are then visualized using the Google Map API. The user can freely zoom in or out and pan. The *TimeStamp* can also be changed to get less recent actions. Besides spatial and temporal filters, a thematic filter is implemented using hastags as known from Twitter. These hastags are automatically extracted from the text body of service offers or requests. The initial GUI is shown in Figure 4.

Next, if users want to submit a request, they may press the plus button depicted in Figure 4 to switch to a request form view; see Figure 5. Users can type in their name, a short title for the service, and a brief textual description which may contain hashtags. The title should be informative, as it will appear on the central browsing view described above. The *Location* (either determined by GPS, WiFi, 3G, or the last cached location) and Unix *TimeStamp* are automatically retrieved from the device and included when the user submits the request to the server.

When browsing the actions on the map, only action titles are displayed. To read the details, users may press the action title and switch to a detailed view as depicted in Figure 6.

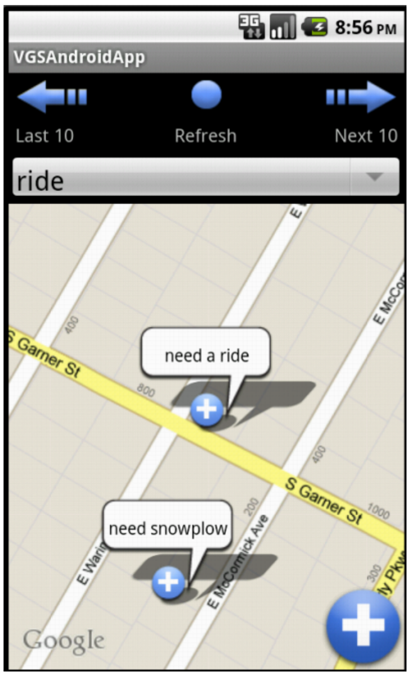


Figure 4: VGS Android GUI to browse and select existing actions

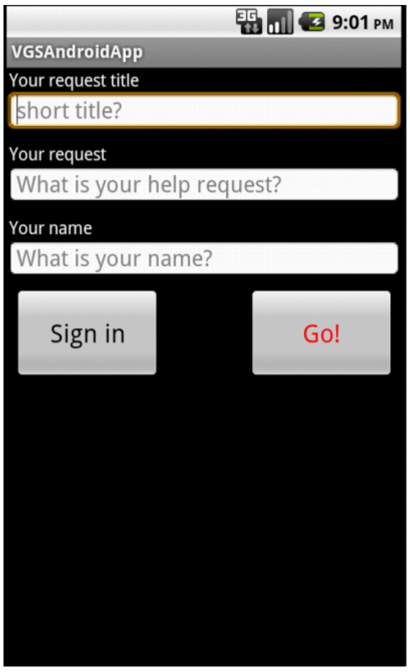
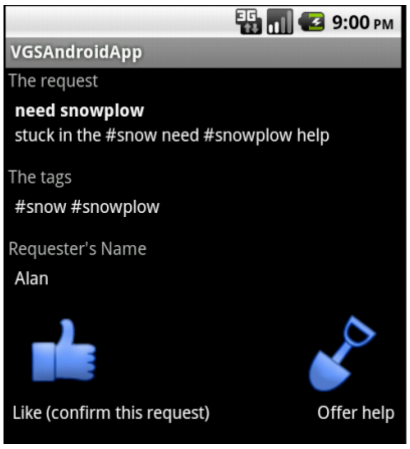


Figure 5: VGS Android GUI to submit a request

In this view, the user may respond to the request by pressing the shovel icon. Responding and submitting requests have similar workflows. Responding adds an additional attribute, *link to original action*. The workflow for both submitting and responding to a request are depicted in Figure 3 as green arrows. The *Like* button is similar to the one in Facebook and confirms a requested services. It can be interpreted in two ways, either the user is requesting the same service (at a nearby location) or confirms the need for this service.


VGSAndroidApp

The request

need snowplow

stuck in the #snow need #snowplow help

The tags

#snow #snowplow

Requester's Name

Alan

Figure 6: VGS Android GUI for detailed action view

In summary, the Triploid API completes the data circle at the client end. The Google Map API and different views (Android's term for content screens) allow the user to interact with the server, e.g., by querying using spatial, temporal, and thematic filters.

5. CONCLUSIONS AND OUTLOOK

In this short paper, we introduced the concept of Volunteered Geographic Services, situated both as an extension of Volunteered Geographic Information (VGI) and in relation to other projects such as SeeClickFix. While two potential scenarios were suggested, the VGS framework itself is not restricted to specific applications but provides a generic infrastructure for storing and retrieving human services. The framework is realized using the Linked Data paradigm and Semantic Web technologies. We also present a mobile client based on Google's Android operating system and deployed on top of the framework. To exchange data between the Android client and the Web service and triplestore, we implemented a marshalling API, called Triploid, on top of the experimental Androjena API. The client implements spatial, temporal, and thematic filters and passes the query over to the Web service using a RESTful approach. The Web service translates the query to SPARQL to retrieve the results from the Jena triplestore. The current state of the software was made available as free and open source software.

The *Mobile Semantic Web* is still in an early stage, the architecture proposed by Tramp et al. [16] and the work from David and Euzenat [3] being rare examples. Consequently, supporting APIs are in an experimental stage, there are no ontologies tailored to the needs of mobile applications, and best practice guides are not available. This makes developing Linked Data driven applications for smartphones challenging. One key problem were missing methods in the Androjena API that have not been ported from Jena so far. Another challenge was the lack of spatial query support in SPARQL. OGC's GeoSPARQL will close this gap and easy development. Based on our first experience with the alpha version of the Parliament triplestore[8], future versions of the VGS framework will be based on Parliament and thus support GeoSPARQL.

[8]http://parliament.semwebcentral.org/

The VGS application and server are still under heavy development; future work will add missing functionality and also focus on interlinking service requests and offers with external Linked Data sources, e.g., data related to places and events. In addition to added functionality and field testing, VGS raises a host of potential problems. First, much like VGI, VGS is potentially vulnerable to both spam and fraud. While studies have examined VGI credibility and reliability both in day to day and crisis uses, the shift from information to action increases both the ability to commit fraud and the potential repercussions of doing so [17, 6, 4]. Trust and reputation are crucial for volunteered services, hence implementation of location-aware trust and reputation models [1] is another direction for further work. In addition to fraud and spam, the VGS system could also be used for illicit purposes. While VGS was conceptualized as an aid to crisis response, the generic framework need not to be put to that use. Trust and reputation models must be tested not only against fraud and spam but also illicit, albeit sincere, service offers. These potential weaknesses are part and parcel of the strength of volunteered, or crowd sourced, information as well. For this reason, VGS is not meant to replace official crisis response services, as their need for verified information and legal accountability cannot be met by a decentralized process, but rather as an on the ground augmentation that incorporates *ground-level* actors, those who are often most involved and affected by emergencies. This said, and as introduced before, VGS is not restricted to emergency scenarios but any services useful for a local community.

Finally, the data produced by VGS users may provide a rich source for spatial and temporal data analysis as well as pattern mining. For instance, the type of requested service will, most likely, differ between neighborhoods and regions. Similarly, one would expect that some service types are correlated to weekdays or seasons.

6. ACKNOWLEDGMENTS

The VGS mobile client and server have been developed as a project in the PSUMobile course in spring 2011 by eleven graduate and undergraduate students from Penn State's Geography program. All source code is available as free and open source software at http://vgs.svn.sourceforge.net/viewvc/vgs/.

7. REFERENCES

[1] M. Bishr and K. Janowicz. Can we trust information? - the case of volunteered geographic information. In *Towards Digital Earth Search Discover and Share Geospatial Data Workshop at Future Internet Symposium*, volume Vol-640. CEUR-WS, 2010.

[2] C. Bizer, T. Heath, and T. Berners-Lee. Linked Data – The Story So Far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.

[3] J. David and J. Euzenat. Linked data from your pocket: The android rdfcontent-provider. volume 6496 of *LNCS*. Springer, 2010.

[4] A. J. Flanagin and M. J. Metzger. The Credibility of Volunteered Geographic Information. *GeoJournal*,

72(3-4):137–148, 2008.

[5] M. Goodchild. Citizens as sensors: the world of volunteered geography. *GeoJournal*, 69(4):211–221, 2007.

[6] M. F. Goodchild and J. A. Glennon. Crowdsourcing geographic information for disaster response: a research frontier. *International Journal of Digital Earth*, 3(3):231–241, 2010.

[7] J. Howe. The Rise of Crowdsourcing. *The Wired Magazine*, 14.06, 2006.

[8] J. Howe. *Crowdsourcing: Why the Power of the Crowd Is Driving the Future of Business.* Crown Business, 2008.

[9] National Research Council. *Successful Response Starts with a Map: Improving Geospatial Support for Disaster Management.* The National Academies Press, 2007.

[10] O. Okolloh. Ushahidi, or 'testimony': Web 2.0 tools for crowdsourcing crisis information. *Participatory Learning and Action*, 59(6):65–70, 2009.

[11] M. Parry. Academics Join Relief Efforts Around the World as Crisis Mappers. *The Cronicle of Higher Education*, March 27, 2011.

[12] M. Raubal and S. Winter. A Spatio-Temporal Model Towards Ad-Hoc Collaborative Decision-Makin. In M. S. M. Painho and H. Pundt, editors, *Geospatial Thinking*, Lecture Notes in Geoinformation and Cartography, pages 279–297. Springer, 2010.

[13] N. Schuurman. An Interview with Michael Goodchild. *Environment and Planning D: Society and Space*, 27(4):573–580, 2009.

[14] A. Sheth, C. Henson, and S. Sahoo. Semantic Sensor Web. *IEEE Internet Computing*, pages 78–83, 2008.

[15] J. Thatcher, C. Mülligann, W. Luo, S. Xu, E. Guidero, and K. Janowicz. Hidden Ontologies - How Mobile Computing affects the Conceptualization of Geographic Space. In *Proceedings of Workshop on Cognitive Engineering for Mobile GIS 2011 (CEMob2011)*, 2011.

[16] S. Tramp, P. Frischmuth, N. Arndt, T. Ermilov, and S. Auer. Weaving a distributed, semantic social network for mobile users. In *Proceedings of the 8th extended semantic web conference on The semantic web: research and applications - Volume Part I*, ESWC'11, pages 200–214, Berlin, Heidelberg, 2011. Springer-Verlag.

[17] M. Zook, M. Graham, T. Shelton, and S. Gorman. Volunteered Geographic Information and Crowdsourcing Disaster Relief: A Case Study of the Haitian Earthquake. *World Medical & Health Policy*, 2(2):231–241, 2010.