Hidden Ontologies – How Mobile Computing Affects the Conceptualization of Geographic Space

Jim Thatcher[1], Christoph Mülligann[2], Wei Luo[1], Sen Xu[1], Elaine Guidero[1], Alexander Savelyev[1], and Krzysztof Janowicz[3]

[1] Pennsylvania State University, USA
{jethatcher|wul132|sux100|guidero}@psu.edu
[2] Institute for Geoinformatics, University of Münster, Germany
cmuelligann@uni-muenster.de
[3] University of California, Santa Barbara, USA
jano@geog.ucsb.edu

Abstract. All software engineering starts with assumptions about the intended purpose, basic functionality, potential users, and their requirements. Developers agree on a shared terminology and workflow, for instance using UML. In fact, they agree on ontological commitments – on an application dependent model of the world. Such commitments, however, are most often driven by business needs and are based on the restrictions of the used platform, e.g., smartphones. What most developers overlook is that (due to the ubiquity of mobile devices and apps) their ontological decisions also affect how users conceptualize geographic space. The role of ontologies in software engineering has been acknowledged and studied before. In this work, based on a *Volunteered Geographic Services* application for Android smartphones, we discuss how specific design choices affect the user's behavior and knowledge. We present first ideas on how such choices can be documented and made explicit.

1 Introduction

In computer science, the term cognitive engineering summarizes those approaches to Human-Computer-Interaction (HCI) that take cognitive processes as blue-prints to improve the usability of software interfaces and mimic human reasoning to ease information retrieval. The increasing ubiquity of computer technology and the Internet led to new research pointing out that HCI is not a one-way street. Cognitive science has studied the influence of tools on human behavior and the conceptualization of our environment well before the rise of the Internet. Recently, human geographers discussed the relation between software code, the conceptualization of space, and capitalism; Kluitenberg argues that *traditional space is being overlaid by electronic networks such as those for mobile telephones and other wireless media* [1, p. 8]. With the increasing popularity of Location-Based Services (LBS), the question of how mobile computing influences the conceptualization of geographic space becomes more urgent. Software

is created for a certain, often monetary, purpose. How do design decisions taken by engineers impact the user's behavior, can we assist developers in understanding the impact of their work and reduce some of the implications?

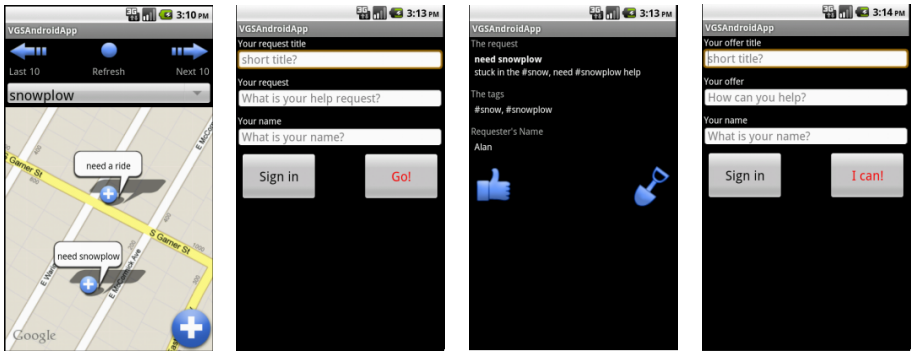
We accepts that all ontological claims involve a *closing off*, i.e., *at once a necessary and problematic condition of knowledge* [1, p.717]. While this view holds for the philosophical definition of ontology, a claim of how the world 'really is', this statement argues it holds equally as true for the computer science definition. Ontologies, *the possible range of meanings offered by an encoded field* [1, p.709], are a necessary component of any programmatic application; however, these programmatic relations are hidden from the end-user. The accompany limitations of epistemology *what it is possible to know* also remain hidden with these 'hidden ontologies'. In this statement of interest, and based on a mobile computing course at Penn State in 2011, we would like to open the discussion of how to understand, quantify, and reduce the impact of such hidden ontologies on the user's conceptualization of geographic space. We try to combine arguments from GIScience with a human geography and ethnography perspective.

2 The Volunteered Geographic Services Use Case

The potential of Volunteered Geographic Information (VGI) have been widely discussed since Goodchild coined the term in 2007 [2]. Platforms such as OpenStreetMap (OSM)[1] or Wikimapia[2] provide means of publishing and requesting geographic data for everyone. That data may not always meet the standards of authority-driven information systems, but it is in many cases more up-to-date, localized, and in some cases even more exhaustive.

We believe that Volunteered Geographic Services (VGS) are a next step. Looking at the massive feedback of, for example, OSM, it appears reasonable to give volunteers a possibility to offer and request not only geographic *information* but also *services*. A geographic service in our terminology is strongly associated with a particular location and performed by human users, and is therefore distinct from the more general service notion on the Web. In that sense it can be treated like any other VGI: it is located in space and time, and for the most part has some attributive data. However, the importance of those properties is different in VGS. Volunteered Geographic Information may have a creation date and a creator, but a user is only interested in location and feature type. In VGS, on the contrary, creation time and creator are of major importance due to concerns about the validity and reliability of the service offer or request. VGS platforms may have different shaping depending on their application, for instance, emergency or carpools. Volunteered Geographic Services may range from simple requests about driving conditions during the winter to emergency scenarios such as rescuing people. Users can request a service using a smartphone App, while other users may offer support.

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



(a) (b) (c) (d)

Fig. 1. Activity screens of the PSUmobile VGS App.

The VGS application developed by the PSUmobile group contains two main components: The server component and the client component.

The VGS server component is an implements a generic framework for handling Volunteered Geographic Services data and can be used for different client applications. It is developed based on Linked Data principles and consists of a triple store, a Linked Data engine responsible for RDF model handling, and a standardized client-server interface. The PostGIS database, Jena API, and the Tomcat platform were used to implement the server. Standardization of client-server communications is achieved by using a VGS ontology which models user interaction in the VGS domain. Basic classes for such interaction include service offers and requests (subclasses of the Action class), and are related through properties such as leadingOffer/leadingRequest and targetOffer/targetRequests. *Leading* is used in analogy to the "like"-functionality of social-bookmarking websites (e.g., "I *also* need a ride"), and *target* signifies the response of an offer to a request (e.g., "I will give you a ride"). In addition, actions have a spatial and temporal signature, a user ID, and a generic message field. The resulting framework is platform-independent, scalable, and remissive of changes to the original Linked Data Model. For instance, an emergency application might introduce an urgency scale as a subclass of Message or an expiration time as a subclass of Timestamp, while still using the VGS generic framework.

The VGS mobile App supports two main tasks: send a request for help, and offer help to a request. Upon opening the application, the user sees the home screen (Figure 1a). From the home screen the two functions diverge. To send a request, the user can press the Create Request button in the lower right hand corner (a blue circle with a plus sign). To offer help, the user presses a request bubble on the map to view the request, and from there, has the ability to decline the request, indicate need for the same service, or offer help to that request.

After pressing the Create Request button, the user is taken to the Send Request form (Figure 1b). It offers three fields: request title, request description, and the user's name. The request description field features a freeform text entry limited to 140 characters, and supports hashtag entry (preceding a keyword with

a #). Pressing the Go! button sends the request to the server with the following information: request text, user name, timestamp, and geographic location (acquired via GPS or 3G/WiFi). This simple two screen navigation supports the rapid publication of requests and offers.

By pressing a request bubble in the map view (see Figure 1a), the user is brought to the View Request screen (Figure 1c). This screen contains information about the requested service. Below the request information, there are two buttons: a Facebook-style like button (the "thumbs-up" hand symbol) and an offer button (the shovel symbol). The like button allows people to confirm or "second" the request – a way of legitimizing the request through repeated confirmation. Pressing the offer button takes the user to a Send Offer form (Figure 1d). This form contains fields for the offer title, offer description, and offerer's name. It has two buttons, a sign in button (with the same functionality as on the Send Request form), and an I can! button that sends the offer to the server with the following information: user-entered information from the form, timestamp, location, and sign-in information (if available).

The home screen also contains query filters to restrict the displayed requests and offers by space, time, or hashtags. Filtering by space is accomplished through zooming and panning the map; the application will only display the most recent n requests in the space defined by the map display (currently n=10 is predefined). The map display serves as a bounding box. The temporal filters are visible at the top: Last 10 (left arrow), Refresh (circle), and Next 10 (left arrow). Last 10 displays the ten previously sent requests, and Next 10 displays the ten more recently sent requests. The Refresh button displays the ten most recent requests. Finally, users can filter by selecting a tag from the drop-down list.

3 Hidden Ontologies in the VGS Use Case

During the creation of the VGS application, the process by which decisions were made was recorded and studied; using a qualitative, ethnographic approach, the team examined how and why decisions were made. One member of the client team undertook a participant observation study. Participating actively in the client design as well as taking exhaustive notes at all meetings, this methodology follows from the work of science studies ethnographers such as Latour or Traweek [3] [4]. The decisions made during the development of the VGS application have direct epistemic ramifications for the end-user as they influenced and shaped the means by which a user comes to know space through the VGS application. The following *vignettes* describe two out of several design decisions and highlight their implications.

Vignette 1: request types and the Mathew Effect During an early meeting of the VGS project client-side team, members suggested that when submitting a request for help, the user be presented with a list of four or five *types* of request (e.g., medical emergency, flood, fire, snow, personal safety). On the request submission form, the user could select a request type from a list auto-populated by the most common types of request over a defined period of time and space. The

design team reasoned that during an emergency many users would be in need of the same type of service. By automatically prompting the user with a series of pre-formatted request types, the application would save valuable time. However, The project lead strongly disliked the idea, citing the Mathew Effect. He reasoned that if the VGS application suggests popular topics. After consideration, the team dropped request types from the user interface design. This decision required a redesign of the request submission form in the UI and the removal of the request type element from the underlying VGS ontology. Left unanswered is whether or not the Mathew Effect would have outweighed the advantages of having the most popular types of request automatically available for selection. We do not know if such a design would save valuable time or instead skew app usage toward conformity rather than utility. What can be said is that the user does not have the ability to see the most common types of request. This suggests that a single ontological decision (whether there are *types of services*) has epistemic ramifications upon what the end-user can know.

Vignette 2: random or recent? When the user starts the application, the phone opens to the home screen (figure 1a) which includes a Google Maps view set by default to zoom level 18. The application sends a request to the server to populate the map of the home screen with service requests. The client and server team discussed which requests the server should send to the client upon application initialization. The head of the client team suggested that the application present the user with ten requests taken from nearby the user's location. These requests would be either the most recent requests sent to the server, or a random choice of requests.

First, the limit of the bounding box presents a default scale at which the user is presented the task of either requesting or offering services. It is up to the user to change the zoom-level to see more requests. If the user does not zoom out or navigate around the greater map area, she may assume that the area covered by the initial map view entirely contains all the requests currently needing a response. This leads to the possibility where a user responds to a trivial request rather than a more urgent request that falls, for example, immediately outside the initial bounding box. This case is not meant to be presented as the norm, but to highlight that necessary default settings affect how the app is used and what users know about their surrounding space.

Second, both the number of requests returned, and whether these requests are displayed because they were recent enough in the request queue, or as a result of a random selection, influences how the user experiences space. The sheer number of requests displayed could easily influence the perceived severity of need surrounding the user. Displaying a set number, whatever that number may be, creates the same initial user experience for a severe crisis as for a set of requests for car rides on a rainy day. The choice between random and time-centric displays also influences how the social space of giving and receiving services is created: during a crisis when many requests are made within the same geographic area requests not rapidly matched with services would begin to disappear from the user's view, although they would remain in the queue. A high volume of requests

in a given area would force a *perpetual present* in which all but the most immediate requests were forgotten. Randomly selecting displayed requests presents other problems as more serious requests could easily be erased for far more trivial situations. Having users rank severity of requests, a potential workaround for this problem which was discussed early on in client team meetings, introduces in the possibility of false inflation of request severity in order to keep own request visible on the map.

4 Making Hidden Ontologies Explicit

While ontology-driven software engineering is offering first tools to create code out of ontologies, the question of how to make design decisions explicit and understand their impact is still an open issue. As illustrated before, each single line of code carries potential ontological commitments, but most of them have no significant influence on the user experience. Which methods can be used to distinguish minor from major decisions, foresee their impact, and provide software engineers with tools to quantify their effect? In the following, we highlight two potential methodologies: an algorithmic and an ethnographic approach and discuss their benefits and limitations.

In previous work, we discussed the problem of assessing the fitness for purpose of an ontology [5]. Domain experts that provide the knowledge to be formalized and the later users have a limited background in knowledge representation and may not understand the logical implications of the used axiomatization. The ontology engineers, in turn, have a limited understanding of the domain. We proposed to use similarity reasoning and implemented a software component to compare the developed ontology to the initial conceptualization of the domain experts and users [5]. We believe that such an approach could be used to study the impact of design decisions as well. The SIM-DL similarity server allows users to rank concepts, such as geographic feature types, by similarity and compare the results to an ontology; where a high correlation indicates conceptual compatibility between the user's conceptualization and the ontology. ConceptVISTA would be another promising tool following a similar approach to semantic negotiation [6]. However, both require formal concept definitions, i.e., the hidden ontologies have to be made explicit beforehand. So far, it is not clear whether all design decisions can be modeled using knowledge representation languages such as the Web Ontology Language OWL and how to combine them with process and component-oriented modeling paradigms used in software engineering. While software engineers may specify some of the core concepts within an ontology, it is unlikely that they will formalize most of their design decisions.

Semantic interoperability is a loaded and difficult to define term that lies at the heart of attempts to understand the embedded meaning of data [7]. Kuhn, for instance, defines semantic interoperability as less a *research topic* than *technical goal* [7, p. 2]. He admits that interoperability must deal with *meaning negotiation and other ways of dealing with organization and social contexts*, but this remains an automated process [7, p. 2]. New algorithms, that measure concep-

tual distance (as discussed above), allow the classification and understanding of meaning, but, at heart, they remain algorithms – meaning is a technical goal [7, p. 15]. Smith and Mark suggest the related concept of *similarity as a relation between instances* defined by cases where *even the slightest further deviation from the norm would imply that they are no longer instances of the given concept*; once more, meaning is measured and quantified, concepts become little more than *topological notion[s]* [8, p. 316].

Other research has followed similar approaches such as semantic proximity or semantic networks [9, 10], but underlying each is an algorithmic approach united by the assumption that semantic interoperability can be automated, and that there is power through the algorithm [10, 9, 11]. While this is a an interesting long-term goal, and research continues in this direction, it is not mere conceptual difference that determines which design decisions are taken and which are not. At the end, programmers are human beings motivated and influenced by their social contexts [12].

As the vignettes illustrate, these decisions are neither good or bad, effective or limiting; they are socially contextual agreements with potential repercussions for the production of and experience in space. From an ethnographic point of view, they cannot be avoided, nor can they be immediately quantified and measured. Participant observation and the interviewing of designers and end-users, thus linking their experiences together, may help to uncover hidden ontologies. Interviews are only possible after design decisions are made and so must be coupled with participant observation of the ethnographer within the design team. It is also unclear how after the fact interviews help during software engineering. Critical geographers have argued that only through empirical, ethnographic investigation may power be decoupled from the algorithm and placed within its socially contingent context. However, from a GIScience perspective, the term *power of algorithm* lacks formal definition and when taken to an extreme suggests a fear of technology.

5 Summary and Outlook

In this statement of interest, we argued that Location-based Services and mobile computing impact how we conceptualize geographic space. While this is not a new insight, we provide a use case that highlights the impact of even simple design decision. These design decisions cannot be avoided during software engineering but must be made in order for code to function. Zook and Graham noted with regards to Google Maps search results that *choice of phrasing and classification can affect visibility and subsequently alter understandings of local geographies* [14, p. 473]. How can we make design decisions and their impact on the user explicit?

To open the discussion, we have highlighted two potential approaches, one from GIScience and one from ethnography, to help making hidden ontologies explicit. Interestingly, while ethnographers may reject purely algorithmic approaches due to their rush to quantify, GIscientists may reject interview-driven

approaches based on the lack of formality. A combination of GIScience methods and Political Geography would be most promising as it would support an inter-subjective quantification without the need to express all involved steps using formal representation languages.

Acknowledgments

The VGS application and server have been developed in a PSUMobile course in 2011 by eleven graduate and undergraduate students from different fields of geography and GIScience at the Pennsylvania State University. While the course started as a technical introduction to Android programming, the students were interested in the cultural implications of LBS as well. All source code is available as free and open source software at http://vgs.svn.sourceforge.net/viewvc/vgs/.

References

1. Kluitenberg, E.: The network of waves: Living and acting in a hybrid space. Open **11**(6) (2006) 6–16
2. Goodchild, M.: Citizens as sensors: the world of volunteered geography. GeoJournal **69**(4) (2007) 211–221
3. Latour, B.: Science in Action: How to Follow Scientists and Engineers Through Society. Harvard University Press, Cambridge, MA (1988)
4. Traweek, S.: Beamtimes and Lifetimes: The World of High Energy Physicists. Harvard University Press, Cambridge, MA (1992)
5. Janowicz, K., Maué, P., Wilkes, M., Schade, S., Scherer, F., Braun, M., Dupke, S., Kuhn, W.: Similarity as a quality indicator in ontology engineering. In Eschenbach, C., Grüninger, M., eds.: 5th International Conference on Formal Ontology in Information Systems. Volume 183., IOS Press (October 2008) 92–105
6. Gahegan, M., Agrawal, R., Jaiswal, A.R., Luo, J., Soon, K.H.: A platform for visualizing and experimenting with measures of semantic similarity in ontologies and concept maps. Transactions in GIS **12**(6) (2008) 713–732
7. Kuhn, W.: Geospatial semantics: Why, of what, and how? Journal on Data Semantics **Spring 2005**(Special Issue on Semantic-based Geographical Information Systems, LNCS 3534) (2005) 1–24
8. Smith, B., Mark, D.: Ontology and geographic kinds. In: Spatial Data Handling Conference. (1998)
9. Schuurman, N., Leszcynski, A.: Ontology-based metadata. Transactions in GIS **10** (2006) 709–726
10. Schuurman, N.: Social perspectives on semantic interoperability: Constraints on a geographical knowledge from a data perspective. Cartographica **47** (2005) 47–61
11. Uprichard, E., R.B., Parker, S.: Geodemographic code and the production of space. Environment and Planning A **41**(12) (2009) 2823 – 2835
12. Thrift, N., French, S.: The Automatic Production of Space. In: Knowing Capitalism. Sage, London, UK (2005)
13. Geertz, C.: The Interpretation of Cultures: Selected Essays. Volume 41. Basic Books, New York, NY (1973)
14. Zook, M., Graham, M.: Mapping digiplace: geocoded internet data and the representation of place. Environment and Planning B **34** (2006) 466–482