Title
Open Location-Oriented Services for the Web

Author
Wilde, Erik

Publication Date
2008-08-01

Open Location-Oriented Services for the Web

Erik Wilde (School of Information, UC Berkeley)

UCB ISchool Report 2008-026
August 2008

Available at http://dret.net/netdret/publications#wil08o

Abstract

Location concepts are still not part of today's Web architecture, which means that applications must rely on higher-level specifications to use and provide location-oriented services. This problem can be approached in two different approaches, the first being a tightly coupled approach for scenarios targeting an integrated system architecture, and the second being a loosely coupled approach, being centered around cooperating services in the open world of the Web. This paper argues that the current specifications for location-oriented services cater mainly for the tightly coupled approach, whereas the loosely coupled approach is not yet addressed by available specifications. A more lightweight and loosely coupled approach to location-oriented services is the central issue for making the valuable data in geographic information systems better available on the Web. Only if location-oriented services can be used easily and cooperatively, today's rapidly evolving infrastructure of wireless data services and mobile devices can take full advantage of these services.

Contents

1 Introduction

There is little doubt that a significant part of the future evolution of the Web will be based on location-orientation of services and devices. The main factor in this development is the increasing availability of mobile devices providing data-based network access. For a long period of time, mobile phone network providers have tried to isolate the mobile phone market from the Internet, introducing technical and financial barriers to maintain control over the services available on mobile devices. Influenced by the iPhone's Web browser and the unlimited data plans associated with the device [16], true mobile access to the Web increased substantially over the last year (since the phone's introduction in June 2007); the next generation iPhone has faster network access and a GPS receiver and will further accelerate the development of the Web as a platform for location-oriented services.

While the landscape of devices (in terms of device mobility and device capabilities) is developing rapidly, there also is a rapidly evolving landscape of Web-based services, which are targeted at users of these devices. Broadly speaking, these services can be divided into to two classes (as shown in Figure 1), one being Web-based user interfaces, and the others being Web-based applications:

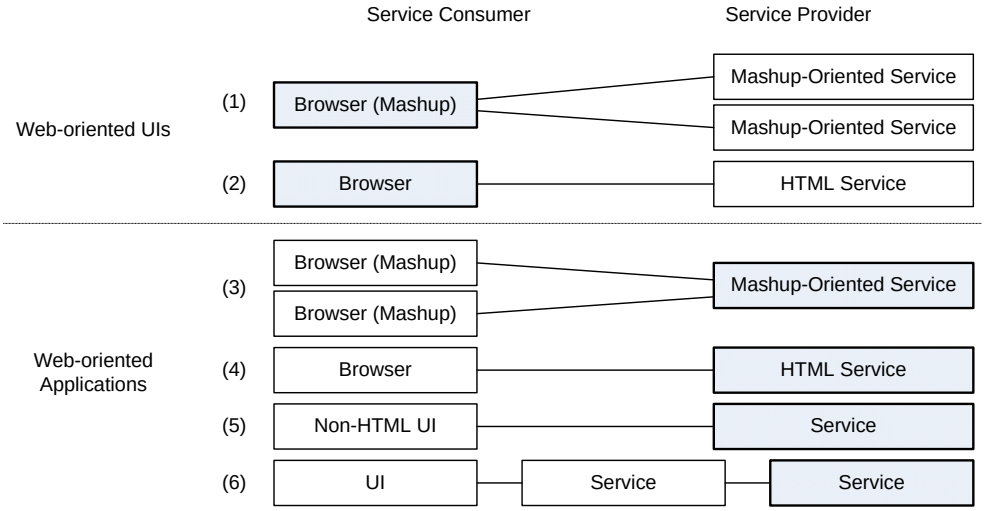

Figure 1: Service Consumers and Providers

- *Web-based UIs:* Web-based UIs are providing a user interface to some location-oriented functionality, they are *service consumers* of some server-side service. Often they are built as mash-ups (Figure 1.1), combining general location-oriented services (such as a map display service) and a more specific service (for example, adding information to the map). Web-based UIs might also only be based on a single back-end (Figure 1.2), in which case this back-end has to provide all the information that is required for driving the UI. If, on the other hand, the UI is implemented as a mash-up, it effectively uses Web-based applications, which are not necessarily intended as standalone applications, but as general services which can be augmented with application-specific information.
- *Web-based Applications:* Web-based applications are *service providers* and can have two different primary scenarios in mind: They can be designed to be used in mash-ups (Figure 1.3), in which case the client is supposed to be script-based (because of this, JSON is often supported as a data format).

> The other scenario does not make such an assumption and provides a Web-based service than can be consumed by any type of client.[1]

For service consumers as well as for service providers, it can be observed that only little standardization efforts so far tackle the issue of how to define and use Web-oriented location-oriented services. The one notable exception are the specifications developed by the *Open Geospatial Consortium (OGC)*, but as explained in more detail in Section 2.1, these services are based on a set of underlying assumptions which are not a good match the evolving landscape of Web-oriented location-oriented services.

The landscape of location-oriented specifications for the Web is in its infancy, as witnessed by the "First International Workshop of Location and the Web (LocWeb 2008)" [3], held at the WWW2008 conference [19], which is the premier Web conference. The idea of a *Locative Web* [40], a Web that makes location explicit in its identification, data representations, and communications, will take some time to gain traction, but first steps are already underway in the form of various draft specifications in some of the key areas of Web technologies (Section 5 gives an overview of the current activities in this area).

What this paper proposes are "Open Location-Oriented Services", based on the idea that only openness can provide the interoperable and collaborative environment in which location-oriented services can flourish. *Openness* in this context refers to the openness of specifications (both technically and legally), so that specifications can be freely used and reused. In a Web-oriented environment, this also requires the specifications to be succinct and modular, so that implementation is not hindered by complex underlying assumptions. Specifically, openness in this context does not refer to what often has been tagged as *WebGIS*, the idea of an open source GIS with a Web-based UI.[2]

We argue that in a service system [37], the implementation of a service is not relevant, and that the important aspect about open location-oriented services is the fact that they are being built around freely available and well-designed services. The implementation of these services is a completely orthogonal issue and should not be confused with the architectural style and the architecture of the service landscape. Section 2 discusses these differences and the importance of architectural styles when designing service landscapes.

2 Service Architecture

The Web's success and particularly the introduction of the *Extensible Markup Language (XML)* [5] in 1998 enabled the transition of an hypermedia system and a system for implementing browser-based UIs, to an era where Web technologies are being used for application-to-application communications. The initially coined term *Web Services* was quickly taken over by a particular set of technologies around the *Simple Object Access Protocol* [4], but it is important to keep in mind that the specific architecture proposed by the SOAP suite of protocols is only one possible approach to building a service-oriented information system based on Web technologies.

Figure 2 shows how information system architectures can be designed, based on the starting point of a *Service Oriented Architecture (SOA)*. Technically, SOA implies little other than identifying and designing service providers and service consumers when building an information system.[3] Essentially, SOA only defines this one constraint, which can be used in different ways to eventually implement an information system. An *Architectural Style* is what guides the design of an information system architecture, and it simply is a set

[1] The three main scenarios for clients are (1) browser-based (Figure 1.4), (2) non-browser UIs running on the client side (Figure 1.5, Section 3 has more information about possible platforms for this architecture), or (3) services which are not (directly) implementing UIs (Figure 1.6).

[2] The idea of a WebGIS essentially is the architecture shown in Figure 1.2 and simply adds a Web-based interface to a traditional standalone GIS. Section 5 contains more details about the relationship between the WebGIS approach and the open services proposed in this paper.

[3] The real challenges of SOA lie in non-technical parts, identifying services and designing them in a way which maximizes flexibility and reuse.

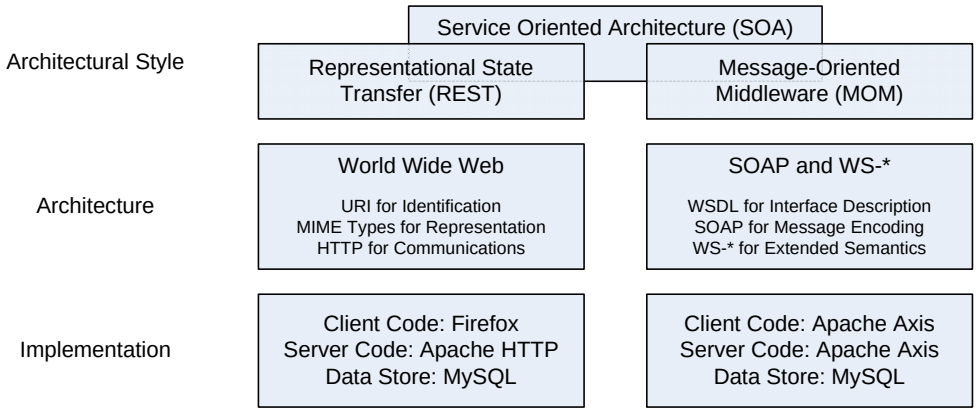

Figure 2: Information System Architectures

of constraints. SOA defines just one such constraint (service orientation as the top-level abstraction), which can be incorporated in different architectural styles.

The two main architectural styles considered for building service-oriented systems are *Representational State Transfer (REST)* [12], the architectural style of the Web, and *Message-Oriented Middleware (MOM)* [20], the architectural style used in many designs of middleware-based IT architectures.[4] It is important to point out that while service orientation as a constraint can be used in both of these architectural styles, the resulting architecture (i.e., the result of applying an architectural style) will look different, because the main abstractions in both styles are different.

The are many discussions about the best architectural style for a given scenario, and REST vs. SOAP debates (even though they oftentimes get confused about the layered model shown in Figure 2, they should be REST vs. MOM debates instead) are being held in many application areas. Currently, only few attempts have been made to compare both styles systematically [36], but regardless of that, it is obvious that, starting with an application scenario, choosing one style or the other will result in different architectures.

One common characterization of the MOM style is that is based on contractual and well-defined relationships, supporting scenarios where peers are willing to invest considerable effort to participate. This is often to referred to as *tight coupling* and is described in more detail in Section 2.1. On the other hand, the REST style puts more emphasis on the ability to cooperate easily, resulting in *loose coupling* which is described in Section 2.2. Both sections illustrate these concepts with examples specific for location-oriented services.

It should be noted that the differences between tight and loose coupling are due to a number of differences concerning various issues, such as service discovery, identification, binding, assumptions about shared data models, interface granularity, state management, and the independent evolution of service providers and consumers. This paper does not give a comprehensive overview of the difference between tightly and loosely coupled scenarios, but instead uses these concepts to differentiate between the two architectural styles discussed earlier.

[4]Newer versions of the SOAP suite of protocols, in particular the *Web Service Description Language (WSDL) 2.0* [7], incorporate support for the REST architectural style as well, but the vast majority of architectures using the SOAP suite of protocols are built based on the style implied in the earlier versions.

2.1 Tight Coupling

Following the MOM architectural style usually results in tight coupling. The assumption underlying this style (and the architecture that is developed when following that style) is that the ultimate goal of an information system architecture is to achieve *integration* (pulling everything together). The trade-offs required for integration are different ones than the ones for *cooperation* (focusing on using things without pulling them together), which is the alternative model emerging from loose coupling (Section 2.2).

Most of the currently available location-oriented services defined by the *Open Geospatial Consortium (OGC)* fall into the category of tightly coupled services. Based on the *Web Services Common Specification* [33], OGC services cover a variety of application areas, available services are the *Web Processing Service* [34], the *Web Feature Service* [31], the *Web Coverage Service* [35], and the *Web Map Service* [29]. Based on the foundations defined by the Common Specification, these services allow applications to exchange information based on an sophisticated shared data model. The Common Specification in its current version has 167 pages and uses ISO 19103 [23] (which itself is based on the *Unified Modeling Language (UML)* [22]) as its notational language.

While this complexity is a result of the inherent complexity of GIS data, it also is a high barrier-to-entry for potential new participants in this service landscape. For the highly specialized exchange of GIS data without any loss of precision or structural richness, such a trade-off between ease or participation and functionality is unavoidable. However, looking at more Web-oriented examples which do not require the full expressive power of GIS back-ends, the question is how to better enable front-ends or intermediary services (as shown in Figure 1.6) to be able to use services provided by GIS back-ends.

Another interesting example of tight coupling is the development of *Location-Based Services (LBS)* in mobile phone networks. As stated in the introduction, mobile phone carriers tried and still try to manage their networks as closed domains, so that they can control which services are made available for mobile phone users. In addition, since the model is based on a closed network, mobile phone carriers can charge LBS providers for services such as disclosing a user's location (these fundamental services are often referred to as *location services* in the realm of mobile phone networking). This approach created tight coupling between mobile phone carriers and LBS providers, for example making it difficult for LBS providers to deploy their applications in different mobile phone networks (because these networks will typically provide different location services).

In an attempt to create a less tight coupling between LBS providers and mobile phone carriers, *OpenGIS Location Services (OpenLS)* [30] were developed. The main idea of these services is to provide a set of location services (the core services are directory, gateway, geocoding, presentation, and routing) which can be used by LBS developers and which will be provided across conforming mobile phone networks. However, this architecture still establishes a tight coupling between LBS providers and mobile phone carriers, whereas a truly loose coupling would work regardless of the network that a service consumer is located in.

It still remains to be seen how much longer the mobile phone carriers can keep up their "walled garden" model of network-specific services and applications. The success of true Web-enabled phones such as the iPhone demonstrates that eventually, mobile phone networks will just become regular data networks.[5] In such a scenario, the client itself must be able to use local services to request location-oriented services, these can either be part of a specific platform (described in Section 3), or in a purely Web-based environment they can be provided through appropriate scripting APIs (described in Section 6).

2.2 Loose Coupling

As mentioned in the previous section, tight coupling is based on the idea of *integration*, accepting contracts and similar commitments which essentially first establish a relationship, and then build an information

[5] Long-range data networks such as WiMax [21] might also contribute to mobile phone networks having a less special role in the future.

architecture based on the assumptions and obligations which govern this relationship. Loose coupling, on the other hand, is built around the idea of *cooperation*, where there is only a minimal set of assumptions that need to be made to enable cooperation between peers. One term frequently used for describing this principle is *Serendipitous Reuse* [38], which describes the fact that it should be easy to cooperate with services, without the need to satisfy elaborate preconditions.

The Web is an excellent example for the benefits (as well as limitations) of loose coupling. Many of the current applications that are often referred to with the terms *Web 2.0* or *mash-ups* are being built by reusing available services in previously unanticipated contexts. This allows the landscape of services on the Web to evolve rapidly. On the other hand, there are two main practical disadvantages of many "loosely coupled" Web 2.0 services:

- *Brittleness:* Because there is no explicit contract, and because requirements change frequently, many Web 2.0 services change their interfaces frequently, often in non-backwards-compatible ways. This causes service consumers to fail if they are not upgraded along with the service provider. This issue is not a fundamental failure of the model, but it shows that the versioning policies of these service providers are not following loose coupling principles.

- *Simplicity:* Because loose coupling should not be based on implicitly shared data models (which oftentimes are complex for non-trivial applications), it is hard to define loosely coupled services for complex application scenarios. Instead, loose coupling is often used for simpler or possible simplified versions of more complex application scenarios. This is particularly true for services which are using the *JavaScript Object Notation (JSON)* [8] instead of XML. While JSON is a good choice for efficiently passing simple data structures to scripting code, it severely limits the ability to use more complex data structures, and to handle this platform-agnostic technologies such as XPath or XSLT.

The goal of loose coupling is to provide better support for cooperation, often at the cost of complexity and integration. For example, while the OGC services listed in the previous section are useful for integration in a GIS environment, they require complex and sophisticated data models to be implemented. This is acceptable in an environment of systems which need this kind of elaborate data model, but is too demanding for the more lightweight approach that is more successful for Web-oriented services.

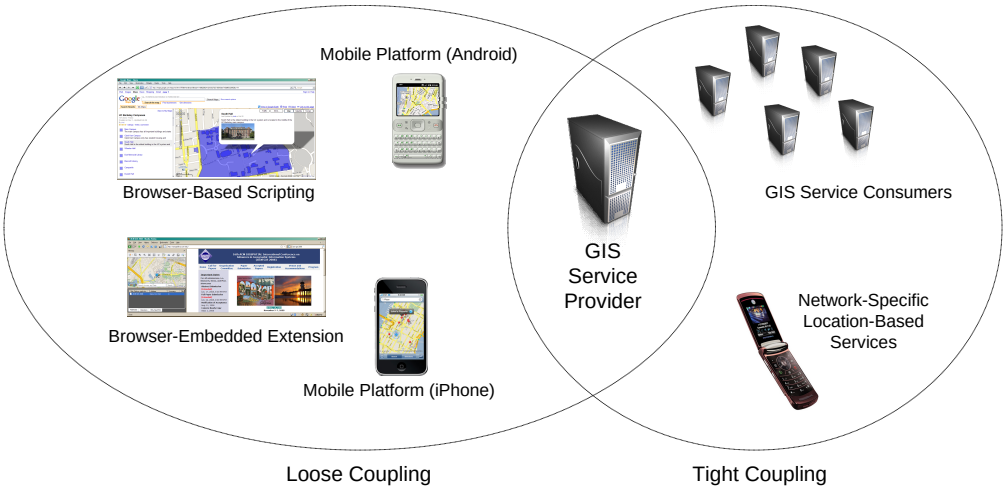

Figure 3: Loosely vs. Tightly Coupled Location-Oriented Services

Figure 3 shows how loosely and tightly coupled services can be regarded as being complementary in a GIS-oriented application scenario. While more sophisticated interactions (typically involving back-end operations, such as exchanging complex data structures) will use more tightly coupled services such as the OGC services presented in Section 2.1, interactions which are more located towards the front-end of a service chain [15] are more likely to be based on simpler data structures. These simpler structures should then be used in simpler services, so that more lightweight service consumers (such as constrained mobile devices) can process them. Section 3 takes a closer look at these differences in service consumers, and how they affect the design of a service (which is described in Section 2).

One aspect that is not explicitly shown in Figure 3 is the fact that the loosely coupled location-oriented services should not be constrained to read-only services. While many location-based services are read-only (for example requesting information about a certain area), the popularity of Web 2.0 applications has demonstrated that *user-generated content* is an important consideration in any Web-oriented scenario [25]. Thus, the loosely coupled service consumers on the left-hand side of Figure 3 should still be able to upload data to the service provider, for example creating user-generated content that is collected while traveling [24].

3 Platforms

Figure 1 show various ways of how service consumers and providers can be structured. One of the important conclusions from that figure is that even when Web-oriented services are considered, it is not necessarily the case that service consumers are always browsers (Figure 1.5 and 1.6). The idea of Web services in general is to use lightweight Web technologies to implement services, but this does only imply the use of Web technologies, not the use of browser-based UIs.

As shown in Figure 3, a well-designed loosely coupled location-oriented service should take various possible configuration into account. Of course there is the *pure browser-based architecture*, where the service is consumed by scripting code running within a Web page, this can be either a mash-up ((Figure 1.1), or a 1:1 association between the browser-based UI and a Web server (Figure 1.2).

Another possible browser-embedded scenario, but this time not based on browser scripting, are *browser-embedded technologies* (also known as *plug-ins*) such as Flash, Silverlight, or Java applets. These technologies run as parts of Web pages, but often also access Web-based services. In this case, the service consumer is still embedded in a Web page, but instead of the cross-browser friendly approach of using scripting, it is using a proprietary runtime environment which must be supported by the browser in order for the Web page to work.

Another possible scenario is that of UIs which are not implemented by Web page scripting (or embedded proprietary code within Web pages). One example is shown in Figure 3, which shows a *browser add-on* (code that is added to the browser by the user, not by a Web page) displaying location-oriented data in a sidebar. In the example shown, it is an add-on[6] using a Web-based map service to provide map display capabilities in a sidebar. This enables users for example to drag&drop addresses (which are then geocoded) into the sidebar and thus map-enables Web pages which do have address information on them, but no embedded map support.

In addition to browser-embedded proprietary code, there can also be *Web-oriented frameworks* such as AIR or Java/JavaFX, which run as standalone application on the client-side, and have good support for Web standards. These frameworks mainly are designed to extend the more limited browser-based client concept with offline capabilities and better support for multimedia. Like virtually all programming environments today, they support all relevant Web technologies and thus can easily use Web-based services.

The Web-oriented frameworks mentioned in the previous paragraph are usually desktop-based, but standalone service consumers can also be *mobile platforms* such as BREW, Symbian/S60, iPhone, J2ME, Win-

[6]Minimap add-on: `http://minimap.spatialviews.com/`

dows Mobile, PalmOS, or Android. All these platforms provide support for consuming Web-oriented services, and thus can be used to implement native location-oriented UIs on mobile devices which are based on Web-based location-oriented services. The location-oriented services which are locally provided on these platforms (such as requesting the current position of the mobile device) currently differ greatly between the various platforms, mainly because location-oriented services are only starting to become a standard part of programming platforms. On the other hand, since all these platforms have built-in support for using Web-based services (supporting core Web standards such as HTTP [13, 14] and XML [6]), a well-designed web-based service can be accessed from all of these devices.

This wide array of platforms described in the previous paragraphs (browsers, plug-ins, add-ons, frameworks, and mobile platforms) demonstrates the range of potential service consumers for location-oriented services. In addition to these, there is the scenario of intermediate service (Figure 1.6), which sometimes also are referred to as *mash-apps*. These can reuse and combine existing services in a similar way to mash-ups (Figure 1.1), but instead of implementing a UI, they provide a service that can be reused.

4 Platforms and Services

One important question for designing Web-oriented services is the question of reuse. It is widely accepted that fundamental technologies such as HTTP and XML should be used, so that implementations can reuse existing tools and code. However, it is of course possible to extend the idea of reuse to other Web-based technologies. One popular example for reuse is the case of *GeoRSS* [32], which is embedding geospatial information into feed formats. By piggybacking on popular feed formats such as RSS and Atom [28], GeoRSS provides a simple but powerful platform for publishing streams of geocoded information items.

The service provided by GeoRSS, however, is a read-only service, which can only be used to publish geocoded information items. Another Web-based technology in the realm of interacting with information items is the *Atom Publishing Protocol (AtomPub)* [17], which extends Atom by also allowing uploads of information items to collections of items. Combining GeoRSS's idea of geocoded information items with AtomPub would yield a service that allowed read and write interactions with geocoded information items.

One of the limitations of such a service, though, is the limitation of GeoRSS's concept of locations. According to the distinction between *spaces* and *places* [18], GeoRSS only supports *spaces* (spatial models of locations). The Web, on the other hand, has shown the power and importance of social concepts, and in terms of location concepts, these are *places*, socially and/or culturally meaningful locations which not may not even have a spatial definition.[7] Consequently, we argue that even though the combination of GeoRSS with other Web technologies would yield interesting results, the underlying model of locations in GeoRSS is too limited for many typical Web-oriented scenarios.

Starting from a seemingly simple use case, though, it becomes apparent that the service design outlined so far has a serious deficiency. If a mobile user requests information items from a location-enabled feed, there is no way how users can specify their location, and by default feeds are time-ordered. This is actually one of the biggest drawback of feed formats: Their roots in news syndication introduced a heavy emphasis on time ordering, so the current specifications assume that feed-published data is primarily time-ordered. However, a new development called the *Feed Item Query Language (FIQL)* [27] is targeted at allowing more specific access to feeds. However, the language is in a very early stage of development and it is not yet clear what features it will support. It could be defined to allow location-specific queries of feeds, so that a mobile client querying a location-oriented feed could specify a location, and the result would be a feed specific for this location.

[7] A typical place would be one that is being used in a social networking system, for examples when users agree to meet "at school." While the system may not even have a spatial definition of this location concept, it is sufficient for identification for all users of the system.

5 Related Work

The service design outlined in the previous section demonstrates that the current landscape of Web technologies offers an array of technologies and ongoing developments which provide a good foundation for open location-oriented services for the Web. In addition to the technologies mentioned already, there are other ongoing developments which are relevant to the intersection between geolocation and Web technologies.

An early approach for bringing geolocation to the Web was an proposal for how to embed geolocation metadata in HTML [9]. This essentially defines a microformat for embedding simple spatial metadata (geolocation and elevation) in HTML. It suffers the same disadvantage as many microformats in that it introduces its own syntax. A more Web-oriented approach would probably use RDFa [1] as its syntax, and the current state of this proposal is that of an expired Internet draft; however, it would be useful to have a standardized way of embedding location information in Web pages.

Another Internet draft defining a core part of Web architecture is one defining a URI scheme for geolocation [26]. While earlier versions were based on geolocation and optional precision, the latest version introduces a tiling model. In both cases, the proposal exclusively focuses on spatial concepts. Ideally, location URI schemes could be used to work with locations as Web-level resources. For example, when activating a location URI, a client could use that as a hint to pass that URI to a navigation application. The recently released Firefox 3 browser, for example, has a mechanism how a URI scheme can be associated with a Web-based service, so clicking on a location URI could take users to their favorite Web-based map service (which would have to be configured in the browser for this scenario to work).

There also is a proposal for embedding geolocation metadata in HTTP[10], which defines new HTTP header fields. The main idea is that location data could be embedded in HTTP, so that HTTP requests already contain the location data that is necessary to provide location-based services. The location model used in this standard is based on geolocation and optionally uncertainty, heading, and speed.

In addition to the three Internet drafts described above there also is a proposal for a DOM API for geo-enabled browsers, which is described in more detail in Section 6. It is worth pointing out that the four proposals described so far (HTML, URI, HTTP, and DOM), which are targeting four main parts of Web architecture, are all based on different location models, and are all based on exclusively spatial models of location.[8]

As mentioned earlier, many discussions around Web-based services for geographic information systems so far have been focusing on UI issues, examples for this are Di Martino et al. [11] discussing the automatic generation of Web-based UIs, and Anderson and Moreno-Sanchez [2] describing the focus on using open software for implementing Web-based GIS architectures.

6 Future Work

As demonstrated by the so far largely uncoordinated efforts described in the previous section, location-oriented services on the Web need a more consolidated foundation, most importantly a location concept which is more advanced than coordinate-based geolocation [39] and can be reused across various parts of Web architecture. At the time or writing, the W3C is in the process of creating a *Geolocation Working Group*, which however will exclusively focus on the specification of the *Geolocation API*. There also may be a *W3C Geolocation Activity* in the mid-term, which would be an ideal place to foster cooperation among all parties interested in turning the Web into a location-aware information system.

One of the interesting questions is whether Atom and AtomPub can be liberated from their time-centric models and will be able to grow into general models of Web-based interaction with collection of items. If

[8]The DOM proposal currently provides geocoding, but this is still under discussions and may change in future versions of the draft.

that can be done, it may well be the case that the most popular method mobile devices communicating with location-oriented services will be through feeds.[9]

There currently still is little cooperation among core Web architecture activities, Internet standardization, the GIS community, and social networking researchers, who also have a lot of contributions to make from their observations of how Web users are interacting and what kind of location-awareness they are using in their interactions. The next two or three years will be a defining period for the way how the Web and GIS will cooperate, and most of the future work required in this area is to look at current social practices, the evolving landscape of mobile devices, location-based services, and to also take a hard look at the issues of privacy and security, which have been completely ignored in this paper, but nonetheless are of utmost importance in the area of location-awareness and mobile devices.

7 Conclusions

This paper describes an approach towards open location-oriented services for the Web. It describes the current state of the art, identifies missing pieces, and proposes a loosely coupled architecture for implementing these services. The next two to three years will see a rapid development of location-aware mobile devices, location-based services targeted at these devices, and the expectations of users to be able to interact with location-oriented services. The current landscape of technologies and proposed technologies covers most of the areas required for the architecture described here, but still leaves something to be desired with regard to taking location concepts one step further than spatial concepts alone, and with regard to generally following a more coordinated approach towards location orientation and reusing core concepts instead of reinventing them in various places.

References

[1] Ben Adida, Mark Birbeck, Shane McCarron, and Steven Pemberton. RDFa in XHTML: Syntax and Processing — A Collection of Attributes and Processing Rules for Extending XHTML to Support RDF. World Wide Web Consortium, Candidate Recommendation CR-rdfa-syntax-20080620, June 2008.

[2] Geoffrey Anderson and Rafael Moreno-Sanchez. Building Web-Based Spatial Information Solutions around Open Specifications and Open Source Software. *Transactions in GIS*, 7(4):447–466, March 2003.

[3] Susanne Boll and Erik Wilde, editors. *Proceedings of the First International Workshop on Location and the Web (LocWeb 2008)*, Beijing, China, April 2008.

[4] Don Box, David Ehnebuske, Gopal Kakivaya, Andrew Layman, Noah Mendelsohn, Henrik Frystyk Nielsen, Satish Thatte, and Dave Winer. Simple Object Access Protocol (SOAP) 1.1. World Wide Web Consortium, Note NOTE-SOAP-20000508, May 2000.

[5] Tim Bray, Jean Paoli, and C. Michael Sperberg-McQueen. Extensible Markup Language (XML) 1.0. World Wide Web Consortium, Recommendation REC-xml-19980210, February 1998.

[6] Tim Bray, Jean Paoli, C. Michael Sperberg-McQueen, Eve Maler, and François Yergeau. Extensible Markup Language (XML) 1.0 (Fifth Edition). World Wide Web Consortium, Proposed Edited Recommendation PER-xml-20080205, February 2008.

[9] Nokia already uses AtomPub in its *LifeBlog* service which allows users of mobile phone to upload multimedia content to a hosting service.

[7] Roberto Chinnici, Jean-Jacques Moreau, Arthur Ryman, and Sanjiva Weerawarana. Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. World Wide Web Consortium, Recommendation REC-wsdl20-20070626, June 2007.

[8] Douglas Crockford. The application/json Media Type for JavaScript Object Notation (JSON). Internet RFC 4627, July 2006.

[9] Andrew Daviel and Felix A. Kägi. Geographic Registration of HTML Documents. Internet Draft draft-daviel-html-geo-tag-08, October 2007.

[10] Andrew Daviel, Felix A. Kägi, and Martin Kofahl. Geographic Extensions for HTTP Transactions. Internet Draft draft-daviel-http-geo-header-05, December 2007.

[11] Sergio Di Martino, Filomena Ferrucci, Luca Paolino, Monica Sebillo, Genny Tortora, Giuseppe Vitiello, and Giuseppe Avagliano. Towards the Automatic Generation of Web GIS. In Hanan Samet, Cyrus Shahabi, and Markus Schneider, editors, *Proceedings of the 15th ACM International Symposium on Geographic Information Systems*, pages 57–64, Seattle, Washington, November 2007. ACM Press.

[12] Roy T. Fielding and Richard N. Taylor. Principled Design of the Modern Web Architecture. *ACM Transactions on Internet Technology*, 2(2):115–150, May 2002.

[13] Roy Thomas Fielding, Jim Gettys, Jeffrey C. Mogul, Henrik Frystyk Nielsen, Larry Masinter, Paul J. Leach, and Tim Berners-Lee. Hypertext Transfer Protocol — HTTP/1.1. Internet RFC 2616, June 1999.

[14] John Franks, Phillip M. Hallam-Baker, Jeffery L. Hostetler, Scott D. Lawrence, Paul J. Leach, Ari Luotonen, and Lawrence C. Stewart. HTTP Authentication: Basic and Digest Access Authentication. Internet RFC 2617, June 1999.

[15] Robert J. Glushko and Lindsay Tabas. Bridging the "Front Stage" and "Back Stage" in Service System Design. In *Proceedings of the 41st Hawaii International Conference on System Sciences*, page 106, Big Island, Hawaii, January 2008. IEEE Computer Society Press.

[16] Greg Goth. Opening the Mobile Net. *IEEE Distributed Systems Online*, 8(11), November 2007.

[17] Joe Gregorio and Bill de Hóra. The Atom Publishing Protocol. Internet RFC 5023, October 2007.

[18] Steve Harrison and Paul Dourish. Re-Place-ing Space: The Roles of Place and Space in Collaborative Systems. In *Proceedings of the ACM 1996 Conference on Computer Supported Cooperative Work*, pages 67–76, Boston, Massachusetts, November 1996. ACM Press.

[19] Jinpeng Huai, Robin Chen, Hsiao-Wuen Hon, Yunhao Liu, Wei-Ying Ma, Andrew Tomkins, and Xiaodong Zhang, editors. *Proceedings of the 17th International World Wide Web Conference*, Beijing, China, April 2008. ACM Press.

[20] Michael N. Huhns and Munindar P. Singh. Service-Oriented Computing: Key Concepts and Principles. *IEEE Internet Computing*, 9(1):75–81, 2005.

[21] Institute of Electrical and Electronics Engineers. IEEE Standard for Local and Metropolitan Area Networks — Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems. IEEE Std 802.16e-2005, 2006.

[22] International Organization for Standardization. Information Technology — Open Distributed Processing — Unified Modeling Language (UML) Version 1.4.2. ISO/IEC 19501:2005, April 2005.

[23] International Organization for Standardization. Geographic Information — Conceptual Schema Language. ISO/IEC 19103:2005, October 2006.

[24] Eric Kansa and Erik Wilde. Tourism, Peer Production, and Location-Based Service Design. In *Proceedings of the 2008 IEEE International Conference on Services Computing*, Honolulu, Hawaii, July 2008.

[25] Ilpo Koskinen. User-Generated Content in Mobile Multimedia: Empirical Evidence from User Studies. In *Proceedings of the 2003 International Conference on Multimedia & Expo*, pages 645–648, Baltimore, Maryland, July 2003. Institute of Electrical and Electronics Engineers.

[26] Alexander Mayrhofer and Christian Spanring. A Uniform Resource Identifier for Geographic Locations ('geo' URI). Internet Draft draft-mayrhofer-geo-uri-02, February 2008.

[27] Mark Nottingham. FIQL: The Feed Item Query Language. Internet Draft draft-nottingham-atompub-fiql-00, December 2007.

[28] Mark Nottingham and Robert Sayre. The Atom Syndication Format. Internet RFC 4287, December 2005.

[29] Open Geospatial Consortium. OGC Web Map Service Interface. OGC 03-109r1, Version 1.3.0, January 2004.

[30] Open Geospatial Consortium. OpenGIS Location Services (OpenLS): Core Services. OGC 05-016, Version 1.1, May 2005.

[31] Open Geospatial Consortium. Web Feature Service Implementation Specification. OGC 04-094, Version 1.1.0, May 2005.

[32] Open Geospatial Consortium. An Introduction to GeoRSS: A Standards Based Approach for Geo-enabling RSS feeds. OGC 06-050r3, July 2006.

[33] Open Geospatial Consortium. OGC Web Services Common Specification. OGC 06-121r3, Version: 1.1.0, February 2007.

[34] Open Geospatial Consortium. OpenGIS Web Processing Service. OGC 05-007r7, Version 1.0.0, June 2007.

[35] Open Geospatial Consortium. Web Coverage Service (WCS) Implementation Standard. OGC 07-067r5, Version: 1.1.2, March 2008.

[36] Cesare Pautasso, Olaf Zimmermann, and Frank Leymann. RESTful Web Services vs. "Big" Web Services: Making the Right Architectural Decision. In Huai et al. [19], pages 805–814.

[37] Jim Spohrer, Paul P. Maglio, John Bailey, and Daniel Gruhl. Steps Toward a Science of Service Systems. *IEEE Computer*, 40(1):71–77, January 2007.

[38] Steve Vinoski. Serendipitous Reuse. *IEEE Internet Computing*, 12(1):84–87, January 2008.

[39] Erik Wilde. Location Management for Mobile Devices. In *Proceedings of the 3rd IEEE Workshop on Advanced Experimental Activities on Wireless Networks & Systems (EXPONWIRELESS 2008)*, Newport Beach, California, June 2008.

[40] Erik Wilde and Martin Kofahl. The Locative Web. In Boll and Wilde [3], pages 1–8.