LogicGeoObject: a Client-Side Architectural Model for Aggregating Geospatial Dynamics from Sensor Web

Xuelin He[1], Jeremy G. Morley[2]

[1] Department of Civil, Environmental & Geomatic Engineering, University College London
ucfsxhe@ucl.ac.uk
[2] Centre for Geospatial Science, University of Nottingham, UK
jeremy.morley@nottingham.ac.uk

Abstract. Sensor technology brings the pervasive capability for observing and communicating real-world dynamics, while current GeoWeb lacks a model for scalable aggregation of client-side logic applications that convey and apply these dynamics. This research explores a client-side architectural model, ADIR (short for Aggregating Dynamics, Interaction & Responsiveness), for organizing geospatial logic applications. A core concept for ADIR is the LogicGeoObject representing a granular unit of geospatial logic. Some inbuilt mechanisms and facilities enable ADIR to cater for geospatially specialized missions. An implementation named LeKML materializes this LogicGeoObject conceptual component. This ADIR architectural model is expected to become a geospatially self-owned, normalized and integral facility for building the mashable Geospatial Logic Web.

Keywords: LogicGeoObject, LeKML, ADIR Architecture, mashability, interoperability, Geospatial dynamics, Proximity-based interaction.

1 Introduction

The Geospatial Sensor Web [1] aims to mirror the real world with multi-source dynamics of interests to provide an integrative context for decision making. Client-side logic is the direct initiator and coordinator of a chained workflow involving back and front ends, so the mashability [2] of client-side logic is a crucial factor in creating an integrative and dynamic front-end for end users.

Most academic and industrial efforts have been concentrating on back-end architectures and techniques. For example, sensor networks observe real-world dynamics, and Open Geospatial Consortium (OGC) Sensor Web Enablement (SWE) services disseminate dynamic sensor data.

There are also certain efforts working on client-side functionality. For example, Tillman & Garnett [3] explore integrated client-side access to OGC Web Services, while the OGC Web Service Access Framework (OX-Framework) developed in the 52°North Community [4] provides a framework for plugging OGC Web Services connectors. In nature, these integrated connectors are all common functional

components to be used by client-side applications. However, existing research is still inadequate to make these client applications mashable and interoperable, i.e., to be able to aggregate various independently developed elements of geospatial logic with arbitrary scalability. It is challenging to mash up heterogeneous sources of applied logic and services [5]. In a scenario based around mashing up various standalone Sensor Web client applications, the potential clash of code logic as well as lack of interoperability is really a headache.

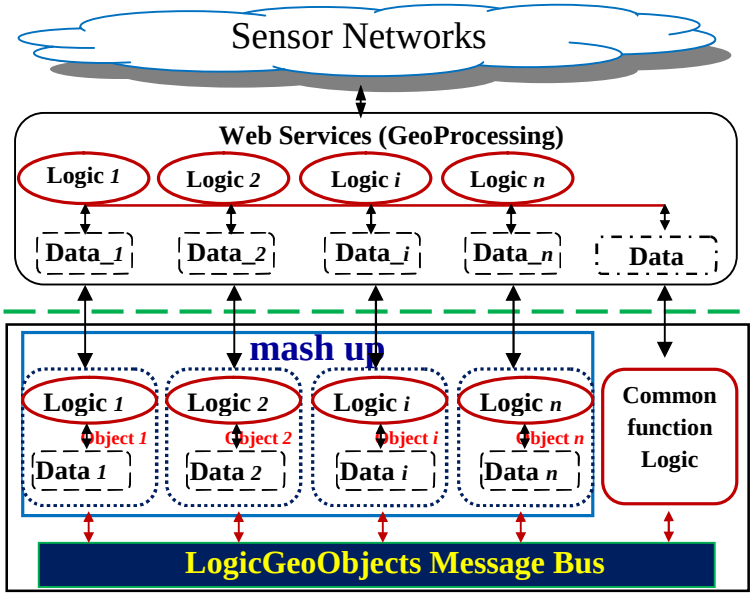


Fig. 1. Evolution from mashing up data into mashing up logic generating and manipulating data dynamically

How can we make heterogeneous sources of geospatial logic mashable and interoperable? This research explores a fundamental solution to equip the GeoWeb with an architectural model on the client side.

2 An Architecture Strategy: LogicGeoObject for Aggregating Dynamics, Interaction & Responsiveness (ADIR)

2.1 Requirements.

To achieve generic mashability, an architecture needs to meet some essential requirements:

a. Mashability. All units of geospatial logic, including those standalone GeoWeb applications, can become mashable ingredients which can generate and manipulate dynamic data respectively.

b. Reusability. Mashable logic units including those crowd-sourced code programs, which are distributed on the Web, can become on-demand code to be dynamically incorporated into new integrative geospatial applications while spatio-temporal context is changing on the fly.

c. Interoperability. All independent units can effectively inter-communicate in a dynamic and loosely-coupled way for scalability, for cooperation to fulfil a certain integrative task, or for incremental development of new applications making use of existing on-demand code logic without necessarily requiring amendment.

d. Portability. Programmed logic units are platform-independent and application-neutral. They can be mashed up into all kinds of geospatial platforms and new applications, such as Standalone Geobrowsers, Web-browser-based mapping applications, or even Desktop-GIS clients.

e. Geospatial suitability and specialization. In-built capability (model, mechanisms and facilities) can cater for Geospatially specialized demands, such as context-awareness, proximity-based communication and cooperation.

2.2 LogicGeoObject: an Architectural Model for Organizing Client-Side Geospatial Logic.

LogicGeoObject for ADIR architecture.

This research proposes an architectural style named ADIR (short for Aggregating dynamics, interaction & responsiveness), within which there is a core concept, Logic-enabled Geospatial Objects (LogicGeoObjects for short). A LogicGeoObject is an autonomous, self-contained unit of data profile and logic, with the granularity of the mashed-up will vary from the whole application scale to individual geospatial features. For example, a LogicGeoObject can refer to a geospatial application (or a module), service, phenomenon, process, incident, activity, functionality, dynamic sensor dataset, geospatial feature–level business or functional logic, etc.

As a result, client-side geospatial sensor applications can be organized as mashable logic units, as illustrated in Fig. 1. Every LogicGeoObject can autonomously access remote Web Services, and can interact with other dynamically aggregated LogicGeoObjects locally via a Message Bus.

Logic-embedded KML: an implementation for LogicGeoObjects.

This conceptual LogicGeoObjects need to be represented by an appropriate data-logic carrier, which is expected to meet the following requirements: format neutrality that is independence from any proprietary platforms or products; potential capability of coupling data (or data profile) with logic; suitable for exchanging geospatial objects; mashability with scalability; Web-based capability for accessing, communicating, deploying and exploiting; etc.

Some media, such as CityGML, KML, etc., are all competent candidates, depending on the client-side environments and tasks. Here we demonstrate an implementation using KML, i.e., Logic-embedded KML (LeKML for short), encapsulating data profile and applied logic. It should be noted that this embedded

logic can deal with other geospatial data types and need not be limited to KML. However KML format is convenient for presenting geospatial data. The <NetworkLink> element in KML can supply an effective mechanism for dynamically aggregating and activating/inactivating on-demand geospatial logic units based on spatial-temporally -aware conditions.

To make this ADIR architecture work via the implementation of LeKML, this research designs a series of LeKML facilities and mechanisms (showed in Fig. 2.), including an approach to couple geospatial data profile and its corresponding applied logic; a unified and platform-neutral programmable Object & Event Model for LeKML; implementing LeKML-supporting capability (LeKML Library) for geospatial platforms, i.e. parsing LeKML, executing logic procedures, and performing in-built LeKML mechanisms and regulations; etc.

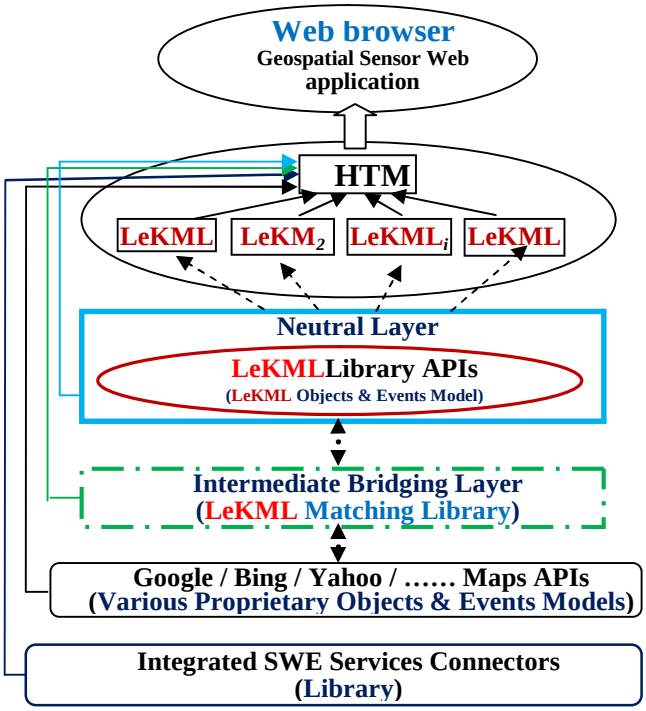


Fig. 2. Platform-neutral LogicGeoObjects:
Object & Event Model for bridging heterogeneities

Runtime LeKML units can work in inter-isolated environments of different executing threads, which can avoid potential clashes between various sources of geospatial logic, and can satisfy parallelism of all LeKML units to process instant updates from respective sensor networks.

Sentience and interoperability for LogicGeoObjects.

In addition to autonomy, mashability and the on-demand assembling mechanism, sentience and interoperability are in-built characteristics for LogicGeoObjects. This LeKML implementation can equip every runtime LogicGeoObject with an internal event pool, and provide a common Message Bus bridging all LogicGeoObjects. A LogicGeoObject can subscribe/listen to certain events of interest to get context-awareness of both its internal state (such as monitoring data mutation) and the state of their surrounding local or remote environment, so as to adapt its behavior and trigger some actions. For example, generally a LogicGeoObjects is designed to only execute (be active) under a specific geospatial conditions such as a certain geographic region and level of detail of views being in the user's viewport currently. So Geospatial context can affect linking, assembling, loading, executing and activating/inactivating LogicGeoObjects.

The Message Bus provides a flexible mechanism for plugging in/out communication to enable interoperability. There is no directly-coupling relationship among these participated LogicGeoObjects. Service discovery and event notification are all based on indirect and anonymous approach via this Message Bus, on which LogicGeoObjects just subscribe to own-interested types of events, and publish own-generated events that may be interest to others. A key Geospatially-specialized characteristic is the location (proximity)-based event-filtering capability built in this Message Bus middleware. An event producer (LogicGeoObject) can specify a certain geographic range of area (bounded by a shape) for constraining event propagation. Only those targeted event consumers (LogicGeoObjects) locating within this area can receive notifications of this event. This kind of event types involves the parameters about valid vicinities. When dispatching events, this Message Bus can analyze these parameters and just pick up those proximity-satisfied LogicGeoObjects (whether stationary or mobile objects) for event delivery. Such proximity-based event-filtering provides a mechanism to dynamically group objects for interoperation, and can reduce the volume of event propagations to improve system performance.

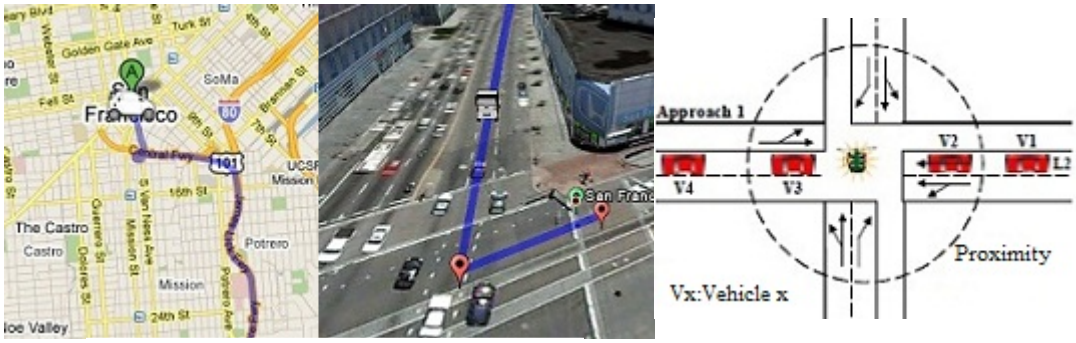
3 Use Case and Scenario

LeKML is a purposely-designed format for a geospatial event-driven architecture catering for responsive granularized client units of Sensor Web to facilitate geospatial digital nervous systems [6]. A LogicGeoObject (implemented using a LeKML file) can function as a smart real-time sensor data feed for dynamically receiving, processing and presenting sensor data. As LogicGeoObjects are autonomous, data-self-monitorable, situation-aware, event-responsive and condition-controlled, these outstanding features make LogicGeoObjects a promising architecturae pattern on building responsive client-side programs for geospatial Sensor Web.

In OGC's SWE framework [7], a SWE client can be appropriately represented by a LogicGeoObject. A LogicGeoObject can act as an autonomous message-interpreting or event-processing engine unit with compact encapsulation of logic (which listens and responds to events) and a data framework (which keeps or stores data parsed from

incoming messages or generated by procedural logic dynamically). This LeKML programming pattern can sensitively monitor, interpret, analyze and process events coming from inside (e.g. internal events such as UI interactions or data changes in this local application) or messages from outside (e.g. external events such as sensor alerts or Web Service Notifications from remote sources).

The mashability of autonomous LeKML files implies that scalable numbers of SWE clients can aggregate within a GeoWeb client application or platform. From a macro perspective, mashing up LogicGeoObjects will imply the aggregation of scalable numbers of event-processing engines and sensor-feeds-consumers into a Web Mapping client, which then implies the capability for the GeoWeb to sense and interact with the real world responsively and instantly. Consequently this ADIR architecture will improve GeoWeb with the capability to browse the physical world with aggregated individual dynamics.



(a) Simulation of automatically driving **(b)** Proximity-filtered cooperation

Fig. 3. Aggregating and correlating LogicGeoObjects by mashing up LeKMLs

This LogicGeoObject model can be used to program various applied logic that retrieve, receive archived or real-time sensor data from the Web, or generate model-simulated data. Fig. 3 demonstrates an example employing this ADIR architecture to aggregate and correlate LogicGeoObjects reflecting dynamics, interoperation and responsiveness. Fig. 3(a) illustrates an original application [8] simulating automatically driving without consideration on traffic control signals, while Fig. 3(b) derives from an example in the COREX [9] project presenting the changes of traffic light state from sensor data. Now these two originally separate and irrelevant applications can be re-programmed with additional cooperative logic using this LogicGeoObject model to fulfill mashability and interoperability. Loose and dynamical correlation and communication can be achieved between the mobile LogicGeoObjects of the vehicles and the stationary LogicGeoObjects of the traffic lights, based on the spatio-temporal context. The Message Bus implemented in this ADIR architecture can use the proximity-based mechanism to filter event propagation, only delivering the events of traffic lights to those vehicles that are currently within a designated proximity.

4 Conclusion

This ADIR with LogicGeoObject is a geospatially-specialized architecture for organizing, communicating and aggregating geospatial logic, applications and services with decentralized granularization and scalable mashability. This architectural model can facilitate the aggregation of real-world dynamics from the geospatial Sensor Web. The LogicGeoObject is a generically-adaptive conceptual model that can be realized via various media carriers such as KML. The LeKML is expected to become a geospatially self-owned, normalized and integral facility for building mashable geospatial Logic Web.

References

1. Van Zyl, T. L., Simonis, I. & Mcferren, G.: The Sensor Web: Systems of Sensor Systems. International Journal of Digital Earth, vol. 2, no. 1, pp. 1-14 (2008)
2. Kazman, R. & Chen, H. M.: The Metropolis Model: a New Logic for Development of Crowdsourced Systems. Communications of the ACM, vol. 52, no. 7, pp. 76-84 (2009)
3. Tillman, S. & Garnett, J.: OGC's OWS Integrated Client Architecture, Design, and Experience. OGC Discussion Paper. OGC Document Number: 05-116 (2006)
4. Broering, A., Jurrens, E. H., Jirka, S. & Stasch, C.: Development of Sensor Web Applications with Open Source Software. In: Proceedings of First Open Source GIS UK Conference (OSGIS 2009)
5. Benslimane, D., Dustdar, S. & Sheth, A.: Services Mashups: the New Generation of Web Applications. IEEE Internet Computing, vol. 12, no. 5, pp. 13-15 (2008)
6. Longueville, B. D., Annoni, A., Schade, S., Ostlaender, N. & Whitmore, C.: Digital Earth's Nervous System for Crisis Events: real-time Sensor Web Enablement of Volunteered Geographic Information. International Journal of Digital Earth, vol. 5, pp. 1-18 (2010)
7. Botts, M., Percivall, G., Reed, C. & Davidson, J.: OGC's Sensor Web Enablement: Overview and High Level Architecture. GeoSensor Networks, pp. 175-190 (2007)
8. Google Code Examples: Driving Simulator, [Online], Available from: <http://earth-api-samples.googlecode.com/svn/trunk/demos/drive-simulator/index.html>.
9. Wu, M.: COREX Project Evaluation Report: CO-operating Real-time sentient objects-Architecture and Experimental evaluation (2004), [Online], Available from: <http://cortex.di.fc.ul.pt/Deliverables/WP4-D13.pdf>.