Standards-Based Geospatial Services Integration for Smart Cities Platforms

Bruno Rienzi, Raquel Sosa, Gastón Abellá, Ana Machado, Daniel Susviela and Laura González
Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Montevideo, Uruguay

Keywords: Geospatial Services, Smart Cities Platforms, Standards.

Abstract: Smart cities usually refer to the use of information and communication technologies to provide citizens with improved city services and quality of life, in an affordable and sustainable way. Geospatial technologies, especially those based on standards, are relevant to this purpose, as location is crucial for organising, processing, and analysing urban information and services. During the last years, many smart cities platforms have emerged to provide support for the design, implementation, deployment, and management of smart cities applications (e.g. FIWARE). Although these platforms frequently consider the spatial dimension, they do not usually provide native support for typical geospatial services (e.g. data access, portrayal, and processing services). Therefore, geospatial-related features provided by applications are usually developed from scratch and on a per-case basis, which leads to code duplication and hinders their implementation agility, maintainability, and reuse. This paper proposes a standards-based solution for geospatial services integration for smart cities platforms, which comprises an overall architecture as well as a reference implementation based on FIWARE and includes three smart cities applications.

1 INTRODUCTION

Smart cities usually refer to the use of Information and Communication Technologies (ICT) to provide citizens with improved city services and quality of life, in an affordable and sustainable way (Santana et al., 2017). Initiatives in the area have tackled different types of city services (Santana et al., 2017), such as transportation, traffic control, air pollution, public safety and waste management, among others.

Geospatial technologies are of paramount relevance towards this purpose, as location is crucial for organising, processing, and analysing urban information and services (Ahad et al., 2020)(Daniel and Doran, 2013)(Sharma et al., 2021). These technologies comprise integrative and analytical technologies (e.g. geographic information systems: GIS), data acquisition and processing technologies (e.g. remote sensing), visualisation and representation technologies (e.g. data models, 3D visualisation), and data management technologies (e.g. relational databases) (de Vries, 2021). Furthermore, geospatial standards are key to easing the access, integration, analysis, and presentation of geospatial data across heterogeneous and distributed computing environments (Lee and Percivall, 2008)(Saborido and Alba, 2020).

During the last years, different software platforms have emerged in order to support the development of smart cities (e.g. FIWARE, InterSCity) (Alberti et al., 2019)(Saborido and Alba, 2020)(Santana et al., 2017). Smart City Platforms (SCP) provide an integrated infrastructure for the design, implementation, deployment, and management of smart cities applications (Santana et al., 2017).

Although SCPs frequently consider the spatial dimension, they do not usually provide native support for typical geospatial services (e.g. data access, portrayal, and processing services). Therefore, the geospatial capabilities of applications are usually developed from scratch and on a per-case basis, which leads to code duplication and hinders their implementation agility, maintainability, and reuse.

This paper proposes a standards-based solution for geospatial services integration for SCPs, which comprises an overall architecture as well as a reference implementation based on FIWARE and includes three applications (i.e. bus detours, beach ranking, and sensor management). The proposal is also aligned with a reference architecture for SCPs proposed by key researchers (Santana et al., 2017).

The rest of the paper is organised as follows. Section 2 presents background. Section 3 describes a mo-

Rienzi, B., Sosa, R., Abellá, G., Machado, A., Susviela, D. and González, L.
Standards-Based Geospatial Services Integration for Smart Cities Platforms.
DOI: 10.5220/0011853500003473
In *Proceedings of the 9th International Conference on Geographical Information Systems Theory, Applications and Management (GISTAM 2023)*, pages 167-175
ISBN: 978-989-758-649-1; ISSN: 2184-500X

tivational scenario, issues, and requirements. Section 4 presents the proposal and Section 5 analyses implementation as well as assessment details. Section 6 analyses related work. Finally, Section 7 presents conclusions and future work.

2 BACKGROUND

This section presents background on smart cities platforms and geospatial standards.

2.1 Smart Cities Platforms

An SCP has been defined as "an integrated middleware environment that supports software developers in designing, implementing, deploying, and managing applications for Smart Cities" (Santana et al., 2017). These platforms aim to ease the development of applications for smart cities by means of services and abstractions (ITU, 2018)(da Silva et al., 2021). They use data from Internet of Thing (IoT) devices and information systems deployed in the city, and provide integration of services promoting the creation of applications to improve citizens' well-being (da Silva et al., 2021).

The unified reference architecture for smart cities platforms proposed by Santana et al. (Santana et al., 2017) identifies key components for SCPs which are described next.

The Cloud and Networking component is responsible for managing and communicating city network nodes. The IoT Middleware is responsible for managing the city IoT network and for the communication with user devices, city sensors, and actuators. The Service Middleware is responsible for managing the services provided by the platform to applications (e.g. monitoring services). The User Management component stores user data and preferences, while the Social Network Gateway enables the platform to integrate with existing social networks.

The Big Data Management module is responsible for managing all the data in the platform and it comprises different components such as: App Repository, which stores applications (e.g. source code); Model Repository, which stores city models (e.g. data models, city maps); Data Repository, which stores data collected from sensors, citizens, and applications; and other components (e.g. for analytics, stream processing, data cleaning, visualization, machine learning).

The architecture also includes security mechanisms and a development toolkit that enable, along with the other components: Developers to implement smart cities applications, Citizens to use these applications, and Managers to govern them.

2.2 Geospatial Standards

The Open Geospatial Consortium (OGC)[1] defines an abstract specification (Percivall, 2020), as the conceptual foundation for geospatial information exchange and interoperability, and over sixty implementation standards that provide the necessary structures and interfaces. OGC also proposes a spatial information framework for smart cities (Percivall, 2015), following a service-oriented architecture and using a subset of their standards. Some key standards from that subset are depicted in Fig. 1, following the OGC Standards Architecture Diagram[2].

The Data Access Services include the Web Feature Service (WFS) (Vretanos, 2010) and the API Features (Portele et al., 2018), which enable a client to query, modify and delete geospatial information. The Portrayal Services include the Web Map Service (WMS) (de la Beaujardiere, 2006) and the Web Map Tile Service (WMTS) (Masó et al., 2010). WMS provides an interface for requesting geo-registered map images in compressed formats (e.g. JPEG). WMTS provides spatially referenced tile images with predefined content, extent, resolution, and coordinate reference system. The Processing Services include the Web Processing Service (WPS) (Mueller and Pross, 2018) and the API Processes (Pross and Vretanos, 2021), which enable a client to request the execution of geospatial processes (e.g. calculating an intersection) either synchronously or asynchronously. These processes are not built-in; they have to be implemented in high-level languages (e.g. Java, Python).

The Data Models and Encodings package includes CityGML (Kolbe et al., 2021), which addresses the representation of virtual 3D city models, and Simple Features Access (SFA) (Herring, 2011), which addresses the representation of 2D vector data. The Discovery package includes standards for geographic information discovery (e.g. using metadata or ontologies). The Sensors package includes standards for modelling, managing, and querying sensors, such as the SensorThings API (Liang et al., 2021) and the Sensor Observation Service (SOS) (Bröring et al., 2012).

[1]https://www.ogc.org

[2]https://www.ogc.org/docs/is

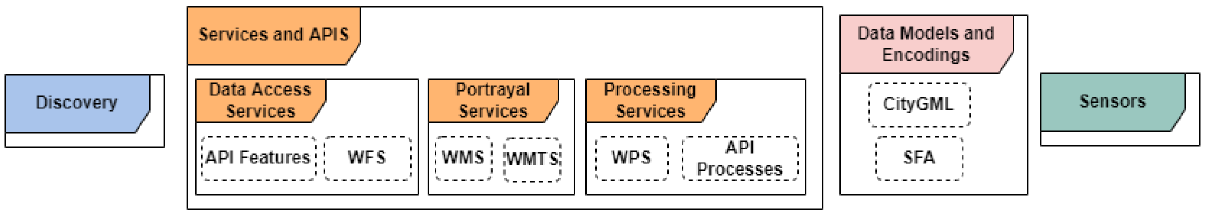

Figure 1: Selected OGC geospatial standards for smart cities.

3 MOTIVATING SCENARIO

This section describes a real-world smart city platform in order to illustrate geospatial issues and requirements within smart cities scenarios.

3.1 Montevideo Smart City Platform and Applications

The Municipality of Montevideo (MM) deploys a FIWARE-based SCP (MM SCP) that receives data from a city-wide IoT sensor network (cf. Fig. 2.). Data are generated by smart things (sensors and private apps) and sent to IoT Agents, which invoke Context Management components: Orion Context Brokers[3] to process the incoming data, Apache Flume to create an intermediate buffer before persisting data, and MongoDB[4] and other databases to persist data.

The Data Viewing and Management applications display real-time and statistical data using dashboards and maps. The Public Applications include the MM Portal and two geospatially-enabled applications that use dynamic maps from server-side geospatial services: the Buses App[5] and the Beaches App[6].

The Buses App shows the real-time location of buses on a map. The location is automatically sent by every bus to the MM SCP. The Beaches App displays information (e.g. current beach flag, algal bloom warnings, etc.) that are daily uploaded to the MM SCP by the staff (e.g. lifeguards) using a private app.

The following limitations are identified in this scenario: (i) each application integrates geospatial and smart things data using its own client-side code; (ii) each application implements geospatial logic with client-side code or server-side (not platform-side) services; (iii) the MM SCP does not provide applications with platform-side, geospatial components.

[3]a Context Broker manages the entire lifecycle of context information including updates, queries, registrations and subscriptions in a smart cities platform

[4]https://www.mongodb.com/

[5]http://www.montevideo.gub.uy/buses/mapaBuses.html

[6]http://montevideo.gub.uy/playas

3.2 Geospatial Issues and Requirements for Smart Cities Platforms

Scenarios like the one described in Section 3.1 present the following issues, which are depicted in Fig. 3.

1. **Lack of an explicit geospatial integration mechanism at the platform level**: Since the SCP does not provide an explicit mechanism to integrate advanced geospatial capabilities, it is up to each application to provide the integration logic between SCP and geospatial logic, adding complexity to each application and potential code duplication among applications.
2. **Lack of complex geospatial logic at the platform level**: Since SCPs only support basic geospatial capabilities, it is up to each application to implement complex geospatial logic, resulting in overly complex, unreusable logic in each application, and potential code duplication among applications.
3. **Sub-optimal use of geospatial standards**: Since geospatial capabilities are not available at the platform level, applications often resort to non-standard Web APIs (e.g. Google Maps API), making standards adoption more difficult than when the platform provides them.

To tackle the described issues, the following requirements are proposed for a Geospatial Smart Cities Platform (GeoSCP).

1. **Provide an Explicit SCP and Geospatial Integration Mechanism:** To facilitate application development, the integration logic must be provided at the platform level rather than at the application level.
2. **Support Standards-based Geospatial Logic:** To facilitate application development and promote code reuse, the GeoSCP must support the incremental deployment of complex geospatial logic at the platform level, avoiding complex geospatial logic at the application level.
3. **Provide a General Architecture that Includes Standards-Based Geospatial Capabilities:** To

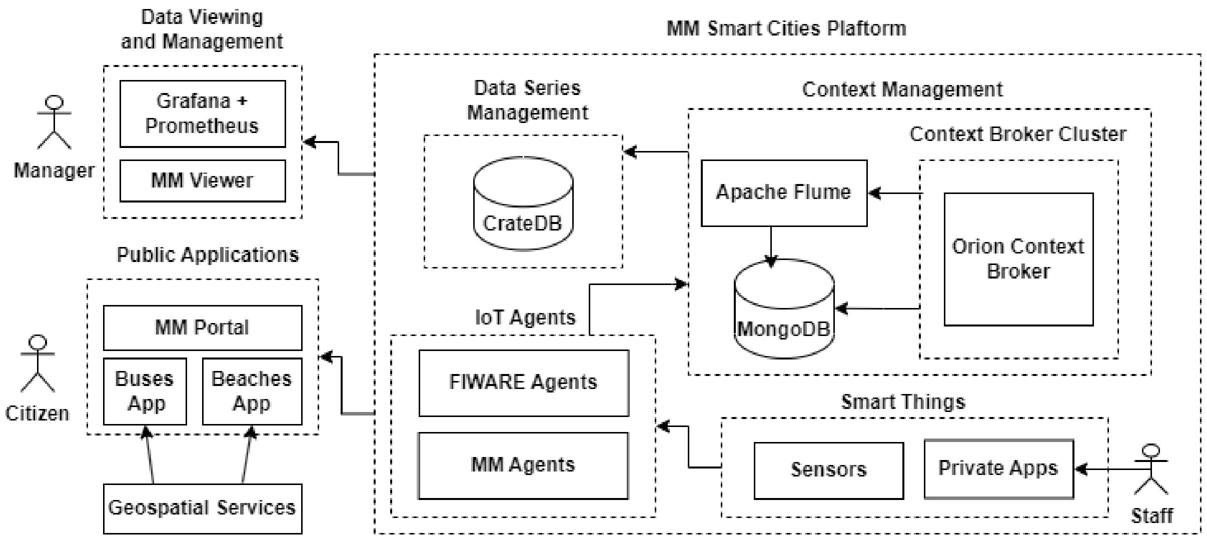

Figure 2: General architecture of MM SCP.

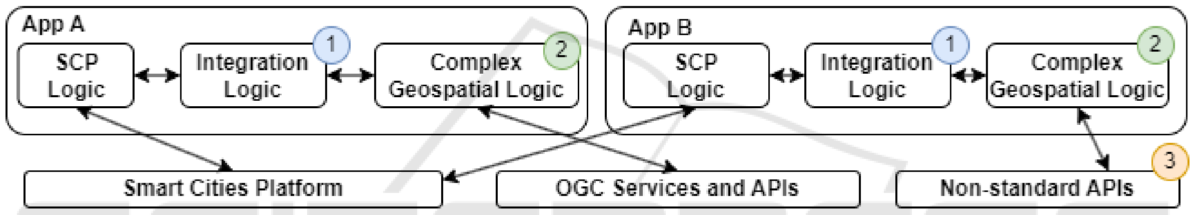

Figure 3: Issues in typical scenarios (numbers correspond to each issue).

prevent the use of geospatial capabilities through non-standard APIs, the GeoSCP must provide the equivalent standards-based capabilities in a well-defined architecture.

In order to facilitate the long-term evolution of the proposal and widen its applicability and interoperability in the scope of other public smart cities initiatives, the following non-functional aspects should be contemplated: (i) low coupling of components to facilitate the substitution of any component; (ii) extensibility of the provided geospatial standards; (iii) feasibility to be implemented with open source software; (iv) alignment to the reference architecture presented in Section 2.1.

4 PROPOSED SOLUTION

This section describes the proposed solution in order to provide a standards-based geospatial services integration for smart cities platforms. A high-level overview of the solution and its architecture is presented, as well as a description of the main components and their interactions.

4.1 General Description and Main Components

The proposed solution is a Geospatial Smart Cities Platform (GeoSCP) that combines a Geospatial Server with an SCP. A Geospatial Server is a specialised middleware that implements OGC standards. More than one geospatial server type may be used to implement all the necessary standards. The general architecture of the solution is shown in Figure 4 and its main components are described in what follows. Other SCP components can be added for specific scenarios.

The Smart Cities Platform in this architecture represents an SCP with no native support for geospatial capabilities. The main components are the IoT Agents and the Context Broker, which are inspired by the FIWARE architecture. The IoT Agents interact with smart things and send data to the Context Broker using a common Context Information Model (CIM) API. Since not every smart thing implements the same transport protocol (e.g. MQTT) or data exchange format (e.g. JSON), a specialised subtype of IoT Agent with a different API for each combination of protocols is needed. The Context Broker acts as the primary access point to context information, using the same CIM

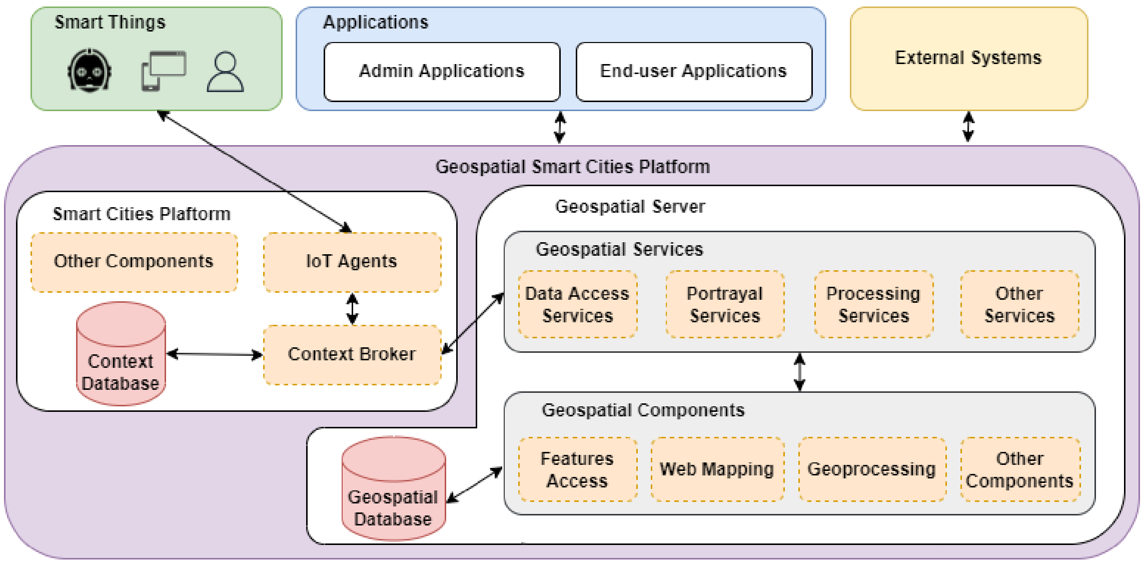

Figure 4: GeoSCP General Architecture.

as the IoT Agents. The Context Broker stores data that is sent by IoT Agents in the Context Database and uses the Publish-Subscribe pattern to send data to subscribers.

The Geospatial Server, which offers an extensible set of services that can be invoked either by applications or by SCP components, is divided into two layers. The Geospatial Services layer consists of Data Access, Portrayal, and Processing services among other services. The Geospatial Components layer comprises components used by the services to execute the business logic. The Features Access component handles CRUD[7] operations on vector data, stored in a Geospatial Database. The Web Mapping component generates map images. The Geoprocessing component provides the executable processes for the Processing services.

The Smart Things component represents all the devices that interact with a corresponding IoT Agent. The Applications component represents the end-user and admin applications that could potentially use the geospatial capabilities of the GeoSCP. The External Systems component groups Web APIs, external Geospatial Services, other SCPs, etc., that could be used by the GeoSCP to complement its internal data or functionalities.

4.2 Components Interactions

This section depicts the most common interaction flows between components.

The **Registration and Subscription Flow**, shown in Figure 5, is performed when a new smart thing is created in the SCP. First, an admin application or an external system invokes the appropriate method (e.g. CreateSmartThing) on the IoT Agent API. At least the smart thing *id* and its *type* must be provided. Next, the IoT Agent invokes a method (i.e. RegisterSmartThing) to register the smart thing on the Context Broker. The IoT Agent API may vary according to its subtype, but the Context Broker always uses the same CIM API, as was explained in Section 4.1. Finally, the admin application or external system creates a new subscription in the Context Broker. The subscription tells the Context Broker that when a smart thing of a certain *type* sends a notification that satisfies a certain *condition*, some geospatial service *endpoint* has to be invoked. For instance, the smart thing could be a bus (type=Bus) that reports certain values like its location, speed, etc. The condition in the subscription could specify that a Data Access Service is invoked each time the bus sends a new location value.

The **Notification Flow**, shown in Figure 6, is performed when a smart thing sends some updated data to the SCP. First, the smart thing invokes the appropriate method (e.g. UpdateValue) on the IoT Agent API. At least, the smart thing *id*, the *attributes* that are being reported (e.g. the location, speed) and the new *values* for those attributes *type* must be provided. Next, the IoT Agent invokes a method (e.g. UpdateContext) to send the updated values to the Context Broker and the Context Broker updates the values belonging to the originating smart thing in the Context Database. At this moment, the Context Broker has to verify that there is a subscription that matches the type (e.g. type=Bus) and that the attributes being updated

[7]CRUD is the acronym of Create, Read, Update, Delete

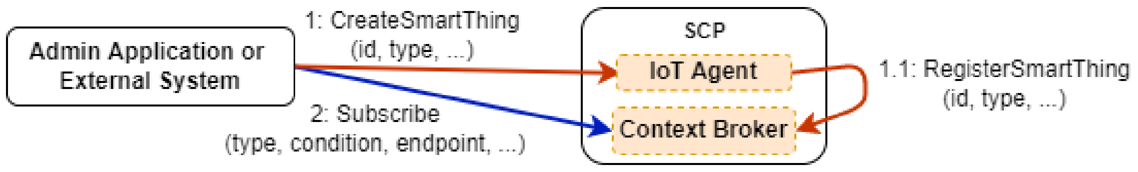

Figure 5: Smart thing registration and subscription creation.

(e.g. location) match the condition of the subscription. If both conditions hold true, the endpoint that was specified with the subscription is invoked (e.g. the GeospatialOperation of a Geospatial Service).

5 IMPLEMENTATION AND ASSESSMENT DETAILS

A prototype of the proposed GeoSCP and three smart cities applications (based on GeoSCP) were developed, which provides a reference implementation for the proposal. This section provides implementation and assessment details.

5.1 Implementation Details

The GeoSCP implementation (Abellá et al., 2021), whose architecture is shown in Fig. 7, is based on FIWARE and includes a Geospatial Server provided by Pygeoapi[8] (Python implementation of the OGC API suite of standards). The OGC APIs (e.g. API Processes) were favoured over the OGC Web Services (e.g. WPS) because they are based on RESTful interfaces with JSON payloads, which goes in line with FIWARE APIs. The Geoprocessing subcomponent is used by the API Processes and includes the supporting processes for the developed applications, but may be reused by other applications. The SFA component is used by the API Features and allows access to 2D vector data (e.g. bus routes) in the PostgreSQL[9] database.

Based on the implemented GeoSCP prototype, three smart cities applications were developed: Bus Detours, Beach Ranking, and Sensor Management[10].

The Sensor Management application manages the sensor's life cycle, using the Registration and Subscription Flow (cf. Section 4.2), and generates simulated data for the other applications, using the platform-side SensorProcess.

The Beach Ranking application let citizens know if a beach is too crowded before arriving. It leverages sensors at beach entrances to detect people coming in and out and uses the platform-side RankingProcess, which receives data from sensors, counts the people at a certain beach, and divides that number by the beach area, generating a crowding index.

The Bus Detours application shows unexpected bus detours (e.g. because of accidents) using the platform-side DetourProcess, which receives the real position of every bus and informs its location as well as route changes in real-time.

The Ranking and Detour processes follow the Notification Flow (cf. Section 4.2) and are published through the API Processes.

In conclusion, the GeoSCP-based applications (Beach Ranking and Bus Detours) show that it is possible to achieve a similar level of functionality than that provided by the MM SCP applications (Busses and Beaches Apps) with the following advantages: (i) the integration between the SCP and the geospatial logic is transparent to the applications, (ii) complex geospatial logic is implemented as platform-side processes, facilitating application development and code reuse and (iii) promoting the use of an unified service interface based on OGC standards.

5.2 Assessment and Discussion

The assessment of the technical feasibility of the solution was carried out through the development of a platform prototype and three applications, which constitute a reference implementation of the solution (cf. Section 5.1).

The three main requirements that were proposed in Section 3.2 are met in the solution. The first one was achieved by leveraging the platform-level Publish-Subscribe mechanism to invoke geospatial services in an event-driven fashion, which prevents applications from implementing the integration logic. The second requirement was achieved by providing standards-based Processing Services to support platform-level, reusable geospatial processes, simplifying the geospatial logic in applications. The third requirement was achieved by integrating a standards-based Geospatial Server with a well-defined architecture of geospatial services and components.

The non-functional requirements were addressed in the following way. The low coupling was achieved

[8]https://pygeoapi.io

[9]https://www.postgresql.org/

[10]Code and screenshots available at: www.fing.edu.uy/owncloud/index.php/s/WxOommkYExin427

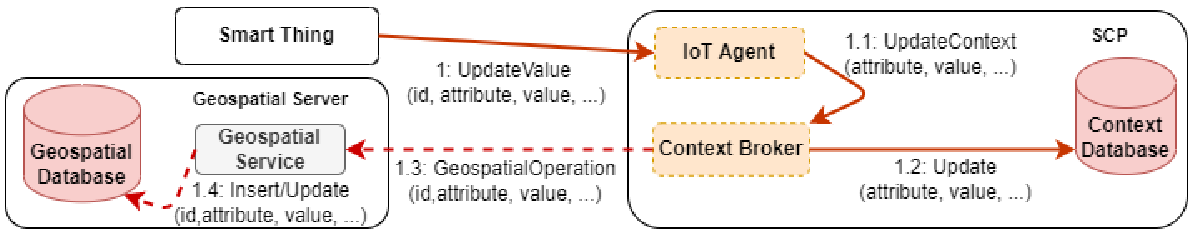

Figure 6: Update values and invoke Geospatial Service.

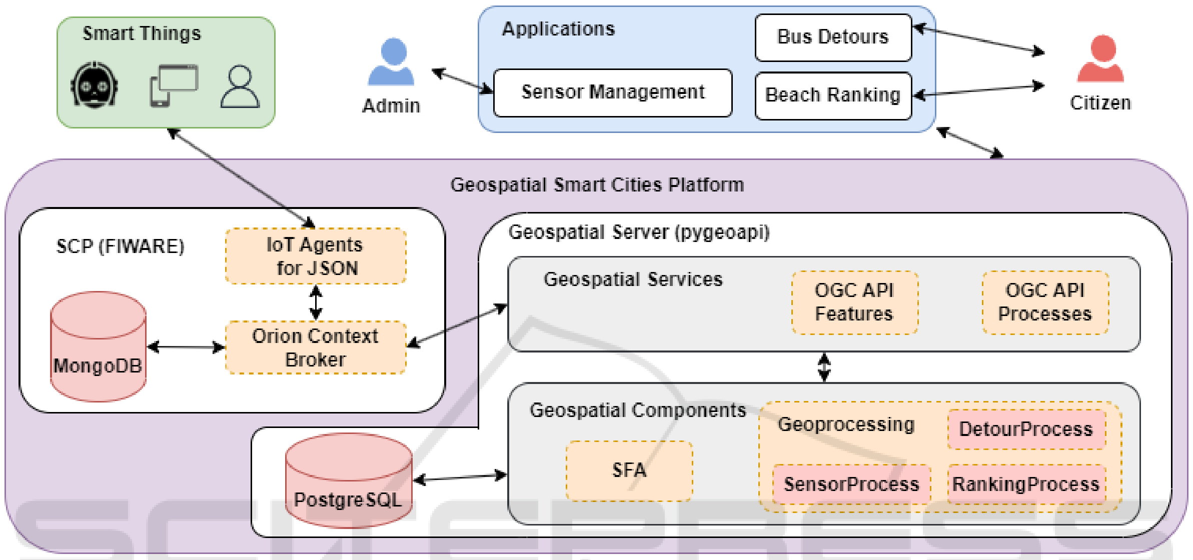

Figure 7: High-level architecture of the implemented solution.

by the use of the Publish-Subscribe mechanism, which makes the inclusion or substitution of geospatial services completely transparent to the other components of the architecture (i.e. SCP, applications, smart things, etc.). By using this mechanism, publishers (e.g. smart things) are loosely coupled to subscribers (e.g. geospatial services) since there are not any direct calls between them (Hohpe and Woolf, 2003). The extensibility of standards was achieved by the inherent extensibility provided by the Geospatial Server and the possibility to add other types of geospatial servers (e.g. GeoServer[11] could be added to support WFS and WPS that are not supported by pygeoapi). The feasibility of a complete implementation based on open-source software was demonstrated with the use of open-source components exclusively (e.g. Orion, pygeoapi). The alignment to the reference architecture is established by the following mapping between component types in each architecture: a Portrayal component is a Visualization component, the Features Access, and Geoprocessing components are Analytics components, the Geospatial Database is a Model Repository, the Geospatial Services are Service Middleware, the ContextBroker and IoT Agents are IoT Middleware; the Context Database is a Data Repository; Bus Detours and Beach Ranking are Citizen Applications, Sensor Management is a Manager application.

[11]https://geoserver.org/

6 RELATED WORK

The integration of geospatial technologies within smart cities solutions has been addressed in different proposals (Bhattacharya and Painho, 2018)(Al-Hader et al., 2009)(Souza et al., 2017)(Yan et al., 2022)(Chaturvedi and Kolbe, 2019).

Souza et al. (Souza et al., 2017) describe the implementation of the Smart Geo Layers middleware and its use in an urban planning application for a smart cities initiative in Natal, Brazil. The main goals of this middleware are to unify and normalise data from various sources, to insert geospatial information related to physical spaces, and to include visualisation as well as data analysis functionalities. Compared to our work, this proposal performs the integration at the data level (not supporting advanced functionalities such as geospatial processes), and it is not based on OGC standards for geospatial services and APIs.

Yan et al. (Yan et al., 2022) propose a Geo Cyber-Physical System Platform for designing smart cities, which leverages cyber-physical systems, GIS, and touchable user interfaces (TUI). The platform provides mechanisms for connecting these three elements, enabling interactive sensing, processing, and actuation in smart city development. Compared to our work, this proposal is broader since it addresses integration with TUI. However, it is neither as detailed as ours regarding the specific geospatial services to provide nor is based on OGC standards.

Bhattacharya et al. (Bhattacharya and Painho, 2018) propose SmaCiSENS: a globally shared open spatial expert system, providing a geo-enabled knowledge based for smart cities. The proposal addresses the integration of interfaces and functions of Spatial Data Infrastructures (SDI) and Sensor Web (SW), as well as sensor data application program interfaces for smart cities. Compared to our work, this proposal also leverages OGC standards but it focuses on the integration of SDI and SW.

Chaturvedi et al. (Chaturvedi and Kolbe, 2019) propose a Web service called InterSensor Service that allows to connect to multiple IoT platforms, simulation data, databases, and simple files to retrieve observations independently of data storage and specific APIs. The service encodes these observations “on-the-fly” according to interfaces such as the OGC SOS and OGC SensorThings API, offering a unified API for the applications. Compared to our work, this proposal does not address the integration of generic geospatial services (e.g. Data Access Services, Processing Services, etc.) into an existing SCP to provide complex geospatial logic to applications, but instead it focuses on providing a standardized interface to applications that can act as SOS and SensorThings clients.

7 CONCLUSIONS AND FUTURE WORK

This paper proposed GeoSCP: a standards-based solution for geospatial services integration for SCPs. The proposal comprises an overall architecture as well as a reference implementation based on FIWARE and includes three applications.

The development of the proposal was driven by requirements of smart cities scenarios, illustrated by a real-world case (i.e. MM SCP), in which SCPs do not provide built-in components to support geospatial capabilities. The proposed solution addressed the identified requirements by: i) providing an explicit SCP and geospatial integration mechanism; ii) supporting standards-based geospatial logic; and iii) providing a general architecture with standards-based geospatial capabilities. In addition, it contemplates non-functional aspects regarding the low coupling of components, geospatial standards extensibility, open-source implementation, and alignment with a reference architecture. The technical feasibility of GeoSCP was assessed through the development of a prototype and three applications.

The main contribution of this work is the overall architecture proposal for integrating standards-based geospatial services with SCPs as well as its reference implementation. These results can be leveraged by smart cities initiatives to enhance their SCPs with geospatial capabilities, to enable and promote implementation agility, maintainability, reuse, and standards adoption for their smart cities applications requiring geospatial support.

Future work includes: i) assessing the proposal with other SCP (e.g. InterSCity) and other geospatial servers (e.g. GeoServer); ii) extending the reference implementation with other geospatial services and applications; and iii) advancing in the validation of the proposal in real-world cases.

ACKNOWLEDGEMENTS

The authors would like to thank the Department of Sustainable and Smart Development of the Municipality of Montevideo for providing us with the required insight into their Smart City Platform which was a key asset for the definition of the motivating scenario.

REFERENCES

Abellá, G., Machado, A., and Susviela, D. (2021). Tecnologías geoespaciales en plataformas de smart cities. Undergraduate thesis, Facultad de Ingeniería, Universidad de la República (Uruguay).

Ahad, M. A., Paiva, S., Tripathi, G., and Feroz, N. (2020). Enabling technologies and sustainable smart cities. *Sustainable Cities and Society*, 61:102301.

Al-Hader, M., Rodzi, A., Sharif, A. R., and Ahmad, N. (2009). Soa of smart city geospatial management. In *2009 Third UKSim European Symposium on Computer Modeling and Simulation*, pages 6–10.

Alberti, A. M., Santos, M. A. S., Souza, R., Da Silva, H. D. L., Carneiro, J. R., Figueiredo, V. A. C., and Rodrigues, J. J. P. C. (2019). Platforms for smart environments and future internet design: A survey. *IEEE Access*, 7:165748–165778.

Bhattacharya, D. and Painho, M. (2018). Location intelligence for augmented smart cities integrating sensor web and spatial data infrastructure (smacisens). In *GISTAM 2018-Proceedings of the 4th International Conference on Geographical Information Systems Theory, Applications and Management*, volume 2018, pages 282–289. SciTePress-Science and Technology Publications.

Bröring, A., Stasch, C., and Echterhoff, J. (2012). OGC Sensor Observation Service Interface Standard. Report, Open Geospatial Consortium.

Chaturvedi, K. and Kolbe, T. H. (2019). Towards establishing cross-platform interoperability for sensors in smart cities. *Sensors*, 19(3).

da Silva, T. P., Batista, T., Lopes, F., Neto, A. R., Delicato, F. C., Pires, P. F., and da Rocha, A. R. (2021). Fog computing platforms for smart city applications - a survey. *ACM Trans. Internet Technol.*

Daniel, S. and Doran, M.-A. (2013). Geosmartcity: Geomatics contribution to the smart city. In *Proceedings of the 14th Annual International Conference on Digital Government Research*, dg.o 13, page 65–71, New York, NY, USA. Association for Computing Machinery.

de la Beaujardiere, J. (2006). OpenGIS Web Map Server Implementation Specification. Report, Open Geospatial Consortium.

de Vries, W. (2021). Trends in the adoption of new geospatial technologies for spatial planning and land management in 2021. *Geoplanning: Journal of Geomatics and Planning*, 8(2):85–98.

Herring, J. (2011). OpenGIS Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture. Report, Open Geospatial Consortium.

Hohpe, G. and Woolf, B. (2003). *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison–Wesley.

ITU (2018). Recommendation ITU-T Y.4201: High-level requirements and reference framework of smart city platforms. Technical report, International Telecommunication Union.

Kolbe, T. H., Kutzner, T., Smyth, C. S., Nagel, C., Roensdorf, C., and Heazel, C. (2021). OGC City Geography Markup Language (CityGML) Part 1: Conceptual Model Standard. Report, Open Geospatial Consortium.

Lee, C. and Percivall, G. (2008). Standards-based computing capabilities for distributed geospatial applications. *Computer*, 41(11):50–57.

Liang, S., Khalafbeigi, T., and van der Schaaf, H. (2021). OGC SensorThings API Part 1: Sensing Version 1.1. Report, Open Geospatial Consortium.

Masó, J., Pomakis, K., and Julià, N. (2010). OpenGIS Web Map Tile Service Implementation Standard. Report, Open Geospatial Consortium.

Mueller, M. and Pross, B. (2018). OGC WPS 2.0.2 Interface Standard. Report, Open Geospatial Consortium.

Percivall, G. (2015). OGC Smart Cities Spatial Information Framework. Report, Open Geospatial Consortium.

Percivall, G. (2020). OGC Abstract Specification Topic 0 - Overview. Report, Open Geospatial Consortium.

Portele, C., Vretanos, P. P. A., and Heazel, C. (2018). OGC API - Features - Part 1: Core. Report, Open Geospatial Consortium.

Pross, B. and Vretanos, P. P. A. (2021). OGC API - Processes - Part 1: Core. Report, Open Geospatial Consortium.

Saborido, R. and Alba, E. (2020). Software systems from smart city vendors. *Cities*, 101:102690.

Santana, E. F. Z., Chaves, A. P., Gerosa, M. A., Kon, F., and Milojicic, D. S. (2017). Software platforms for smart cities: Concepts, requirements, challenges, and a unified reference architecture. *ACM Comput. Surv.*, 50(6).

Sharma, P., Singh, R., and Srivastava, A. (2021). *Analyzing the Role of Geospatial Technology in Smart City Development*, pages 1–20. Springer International Publishing, Cham.

Souza, A., Pereira, J., Oliveira, J., Trindade, C., Cavalcante, E., Cacho, N., Batista, T., and Lopes, F. (2017). A data integration approach for smart cities: The case of natal. In *2017 International Smart Cities Conference (ISC2)*, pages 1–6.

Vretanos, P. P. A. (2010). OpenGIS Web Feature Service 2.0 Interface Standard. Report, Open Geospatial Consortium.

Yan, W., Kiyoki, Y., and Murakami, Y. (2022). *The Geo CPS Platform for Designing Smart Cities.*