Towards a Semantic Message-driven Microservice Platform for Geospatial and Sensor Data

Matthias Wauer[1] and Axel-Cyrille Ngonga Ngomo[2]

[1] Universität Leipzig, AKSW Group
Augustusplatz 10, 04109 Leipzig, Germany
wauer@informatik.uni-leipzig.de
[2] Universität Paderborn, Data Science Group
Warburger Straße 100, 33098 Paderborn, Germany
axel.ngonga@upb.de

Abstract. Many applications of cyberphysical systems rely on the large-scale integration of geospatial data and sensor data. These types of data are both characterised by their heterogeneous structure, different formats for their representation, and their enormous volume. In recent years, the Semantic Web community has developed vocabularies and standards for representing and querying geospatial and sensor data. In this paper, we present an ongoing research project addressing the need for a flexible and scalable platform of services handling such data. The platform comprises services for the extraction, transformation, interlinking, fusion, quality assurance, and delivery of smart data. In contrast to related work, we pursue a message-driven approach for a flexible and scalable integration of microservices. We further discuss several use cases which motivate the design of our system architecture.

Keywords: service platform, message-driven, geospatial data, sensor data

1 Introduction

An increasing number of applications depend on the flexible integration of live data, such as sensor data. Most sensor data is inherently related to geospatial data, e.g., the location of a certain measurement or place references in metadata. Other sources, like social media feeds, can contain place names that are valuable for real-time applications.

Despite the advances in research and technologies for the semantic integration of geospatial data, it is still challenging to build respective applications. It is particularly difficult to utilise live streams of data with geospatial aspects, such as sensor data, in real-time applications.

The difficulty is mostly caused by three aspects. First, components for translating data into RDF, analysis, interlinking and enrichment have to work with typically small but frequent payloads of data streams. Second, the communication between these components needs to be flexible and reliable. Third, the entire system has to scale dynamically according to a changing load of messages, which typically requires a distributed implementation.

Examples of respective applications include routing, geomarketing, and all sorts of analytics, such as business intelligence and predictive maintenance. Internal case studies for the use cases discussed in Section 2.1 indicate that these can be seen as a workflow based on reusable components for different process phases. Various data streams are provided as input. The data of interest is extracted and transformed into a common format, interlinked and fused with related data. The output might be examined with quality metrics before it gets stored and visualised.

There are many tools available already for handling these general tasks with Semantic Web technologies. However, hardly any of them support the specific needs of real-time geospatial data streams. Also, there is no framework that provides a runtime environment for the integration of such tools and addressing additional requirements.

In this paper, we present an ongoing research project focusing on the following primary research question: How to design a platform for flexible integration of different geospatial and sensor data (Section 2)? We propose a microservice-based message-driven approach, validate it (Section 3) and contrast it to related work (Section 4).

2 The GEISER Project

Based on a range of use cases and corresponding requirements, we derive objectives and an architecture for the GEISER project. Then, we describe the current implementation of the GEISER platform.

2.1 Use Cases

We develop and evaluate the GEISER platform along the following three primary use cases.

Data-driven geomarketing. With more than 500 billion euros in retail revenue in Germany alone,[3] local retail and catering are important employers and taxpayers. To be successful in a competitive market, regional companies have to adapt their products, services, marketing strategy, opening hours and special offers to their audience. Data-driven geomarketing would already be possible today using all available sources: the Web, social media and news, message boards, mobility and cellular network data, open data, their own customer data, etc. However, these sources are monolithic and lack explicit geospatial references.

In GEISER, we want to build a geomarketing assistant application which informs companies in near real-time considering all relevant information. For

[3] Source: `https://de.statista.com/themen/136/einzelhandel-in-deutschland/`

example, a local restaurant would be alerted about a band playing nearby, extracted from social media and event data (e.g., Twitter), so it could provide special offers to their fans. Further requirements include predictions based on weather warnings and forecasts (data from DWD.de and openweathermap.org) and public transport (via data.deutschebahn.com).

Intelligent parking. Roadside parking is challenging in crowded city centres. While personal navigation assistants (PNA) can guide users to a certain destination, they are not aware of available parking spaces on that road. Existing solutions to that problem require infrastructure support, so they only work in locations where sensors installed in the road already identify cars.

In GEISER, the messages of connected PNA are used to detect where cars have started and finished their journeys. Based on that data, it is possible to determine the probability for finding a free parking space on a given section of a road at a certain time. Instead of driving to a destination with a low chance of finding a parking space, the PNA would provide approximate guidance through nearby roads with a higher probability.

This probabilistic routing can be enhanced with more detailed information on how many people frequent a region at a certain point in time, and why they do so. Therefore, we need to interlink and interpret different data sources, including the weather data discussed above, local events (from Twitter) as well as floating car data and traffic incidents from TomTom. We build an extended parking search service based on the GEISER platform and On Street Parking.[4]

Predictive maintenance and industrial service. Industrial service is becoming increasingly important. Sensor networks in industrial appliances generate huge amounts of data that can be used to predict a device failure based on historical information. Thus, industrial service can be planned and executed even before a failure occurs, reducing the cost of a halted manufacturing process.

In GEISER, we plan to link the predictive maintenance with service technician scheduling and spare part logistics. For an optimal service coverage, many other data sources with geospatial aspects have to be linked to industrial sensor data. Goals include a less unnecessary travelling and improved industrial service quality. Requirements include the integration of service technician deployment planning[5] with weather and traffic data sources discussed in the other use cases.

2.2 Goal and Objective

As shown in the use cases above, there are many applications that require flexible processing of data streams from different sources with geospatial or temporal references. However, these relations are rarely inherent to the data or otherwise available in a machine-processable way.

[4] `https://developer.tomtom.com/on-street-parking/documentation`

[5] Data coming from the STEP project: `https://www.projekt-step.de/en/`

Therefore, it is the primary goal of the GEISER platform to enable mass processing of geospatial and sensor data with the following aspects:

1. Data extraction recognising the references between data and known entities
2. Semantic technologies for adding these references to the data
3. Flexible microservice architecture and asynchronous messaging for a scalable foundation of different applications

2.3 Architecture

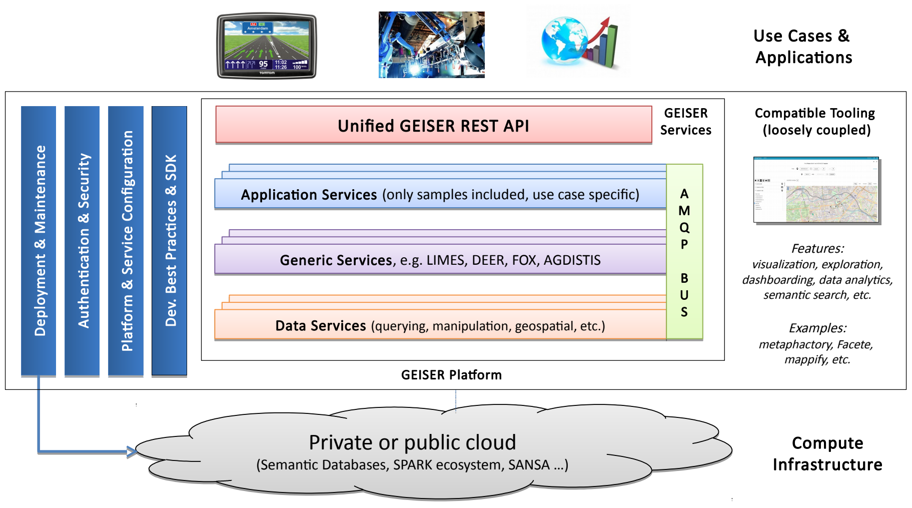
Use Cases & Applications
Unified GEISER REST API
GEISER Services
Compatible Tooling (loosely coupled)
Deployment & Maintenance
Authentication & Security
Platform & Service Configuration
Dev. Best Practices & SDK
Application Services (only samples included, use case specific)
Generic Services, e.g. LIMES, DEER, FOX, AGDISTIS
Data Services (querying, manipulation, geospatial, etc.)
AMQP BUS
Features: visualization, exploration, dashboarding, data analytics, semantic search, etc.
Examples: metaphactory, Facete, mappify, etc.
GEISER Platform
Private or public cloud
(Semantic Databases, SPARK ecosystem, SANSA ...)
Compute Infrastructure

Fig. 1. The GEISER architecture.

The GEISER platform is comprised of several horizontal and vertical layers, as shown in Figure 1.

Three horizontal layers divide the lower and higher level services available in the platform. The lower layer consists of data access and handling services, such as extraction services and triple store access. The middle layer contains generic platform services for processing semantic data, such as quality assessment, interlinking, and fusion services. The upper layer contains usually domain-specific higher level application services. Examples include the probabilistic routing service for the parking use case and a target audience estimation service for geomarketing.

All of these services can access each other via a message bus. We chose the RabbitMQ implementation of the AMQP standard. This enables different message exchange patterns, including reliable asynchronous messaging with delivery guarantee. In addition to the flexibility of the message routing, which can easily be modified at runtime, this brokered approach was favored to a much more limited, entirely REST-based implementation. We also favored RabbitMQ over Apache Kafka, primarily because of simpler consumer implementation, support for the MQTT protocol supported by many sensors, and wider support for different messaging patterns such as RPC [1].

The vertical layers provide the platform support for cross-cutting concerns like authentication, deployment, and service configuration. Application access is enabled by a REST API on top of the service layers. This allows external applications, such as the routing function in a PNA, to invoke services in the platform and receive results without having to be a message bus endpoint. Additional tools provide further functionality, including visualisation and semantic search.

2.4 Implementation

Since the project is still in an early phase, we have only started to implement and validate our design decisions.

In contrast to existing semantic service infrastructures, which typically utilize HTTP requests for service interactions, we have implemented an asynchronous messaging approach. An AMQP topic exchange can be applied very flexibly to implement different messaging patterns, including publish/subscribe-like interactions. These can also be modified at runtime, enabling ad-hoc changes to the processing of messages.

Additionally, GEISER proposes a message passing approach between a linear chain of services. For each use case, messages flow through a list of components. In order to specify the respective services, we store their topic identifiers in an AMQP header field. This is usually configured for the service sending an original event to the message bus. The remaining services are encoded as $\texttt{service}_1\texttt{.service}_2\texttt{.\{...\}.service}_n$ in the AMQP routing key. Consequently, each service is aware of the intended following routing. The following example shows a JSON-LD message from an interaction in a chain of services. The FOX service has performed Named Entity Recognition (NER) on the given text and sends the results to the following service `location` which annotates the geocoordinates of the entity.

Listing 1. Message processed by the entity recognition service fox, to be processed by further location and store services. Note: lines starting with two slashes are basic properties of an AMQP message, not part of the JSON-LD message content

```
// routing key: location.store
// message-id: 2
// correlation-id: 123
// app-id: fox
// reply-to: annotation
// content-type: application/ld+json
// content-encoding: utf-8
// type: LocationRequest
// header:
```

```
//  [services: ["annotation", "fox", "location", "store"]]
{
 "@context": "http://example.org/context/fox.jsonld",
 "text": "Leipzig is a beautiful city.",
 "fox:": [{
  uri: "http://dbpedia.org/resource/Leipzig",
  name: "Leipzig"
 }]
}
```

Many services are already available with a REST API, but don't feature AMQP support. In order to provide a simple option for integrating them, we have developed a wrapper approach for translating AMQP messages into REST requests and vice versa, based on a declarative service description.[6] The current implementation is based on NodeJS and Go.

Furthermore, we implemented a small library based on Spring AMQP for implementing the message bus interfaces for Java components. The following listing shows that very little code has to be implemented for a GEISER message handler. The `handleRequest` method is invoked with the unmarshalled request and the original AMQP message. Besides the `@RabbitListener` annotation specifying the topic this service is listening to, the library provides request (un-)marshalling and extracting the next routing key for the response message.

Listing 2. Stub for implementing a GEISER service interface in Spring Boot

```
/* imports */
@Component
public class MyService {

  public static final String ROUTING_KEY = "myservice-v1.#";
  public static final String QUEUE_NAME = "myservice-v1";
  public static final String EXCHANGE = "geiser";

  @RabbitListener(bindings = @QueueBinding(key =
      ROUTING_KEY, exchange = @Exchange(type =
      ExchangeTypes.TOPIC, value = EXCHANGE, durable =
      "true", autoDelete = "true"), value =
      @Queue(autoDelete = "true", value = QUEUE_NAME)))
  public void handleRequest(Request request, @Payload
      Message message) throws IOException {

    /* handle request */

    String nextRoutingKey =
        ServiceUtils.nextRoutingKey(message);
    rabbitTemplate.send(EXCHANGE, nextRoutingKey,
        MessageBuilder...build());
  }
}
```

[6] https://github.com/AKSW/micropipe-proxy

For monitoring an exchange we can listen to all topics of an exchange with a wildcard and store the messages, e.g., into a triple store. In contrast to storing the entire message as a literal, we decided to store each AMQP value as separate properties to be able to effectively search that log using SPARQL. We realised that many properties that we had modelled for messages following a requirements analysis phase are already available as basic properties in AMQP. We only had to specify the details of respective values, e.g., how services are referenced in the `app-id` property representing the origin service of a message. Since to the best of our knowledge no such thing exists, we created a basic RDF vocabulary[7] for these AMQP properties.

For managing the deployment of an application consisting of services we use Docker and Docker Compose. Each service is dockerized and configurable, e.g., using environment variables. The set of services required for an entire application workflow is configured in a compose configuration file.[8] The following listing shows part of such a file for the geomarketing use case. When deployed using `docker-compose up`, the Twitter feed service will push new tweet messages mentioning Leipzig onto the message bus, which will subsequently be translated, annotated (by FOX) and stored.

Listing 3. Docker Compose configuration (partially) for a geomarketing use case

```
services:

  geiser-twitter-feed:
    image: mwauer/geiser-twitter-feed
    links:
      - rabbit:rabbit
    depends_on:
      - rabbit
    environment:
      - keywords=leipzig
      - routing=translate.fox.store
[...]
  translate:
    image: mwauer/geiser-translate-service
    links:
      - rabbit:rabbit
    depends_on:
      - rabbit
    environment:
      - translate_input_attribute=sioc:content\[]\@value

  rabbit:
    image: rabbitmq:management
    ports:
      - 5672:5672
```

[7] `https://github.com/mwauer/geiser/tree/master/messaging-ontology`

[8] Further schema information: `https://docs.docker.com/compose/compose-file/`

So far we have implemented a set of generic services for reading and writing RDF, interlinking (using LIMES), NER and relation extraction (using FOX), enrichment and fusion (using DEER) and machine translation (using the Yandex API). Further custom services are being developed for the respective use cases, e.g., a probabilistic routing service.

While this implementation appears to be sufficient for the presented use cases, we are currently evaluating whether more complex workflow patterns are necessary (see Section 3). However, these could easily be achieved with a microservice inspecting the incoming message and potentially modifying the routing key. Thus, conditional workflows can be implemented.

3 Validation

Due to the early stage of the implementation, we have focused on the functional evaluation of the platform and performance measurements of the components, according to the requirements gathered from the use cases. These are primarily related to data sources (various types of static and real-time data) and required services for processing and querying data.

We have successfully run an instance of the service composition shown in Listing 3, showing the applicability of the approach for a part of data-driven geomarketing use case described in Section 2.1. Further functional evaluation depends on the availability of further services specific to the use cases, which are currently being developed.

With regards to the extraction of data from unstructured sources, the evaluation of FOX in the OKE challenge [13] shows that it outperforms comparable NER implementations. For the semi-automatic extraction of RDF from tables, TAIPAN [2] shows significant improvements compared to the state of the art. With regards to interlinking and fusion on geospatial data, RADON [11] as part of the LIMES component was shown to be on average 65 and 834 times faster than SILK for single-threaded and parallel execution, respectively.

4 Related Work

4.1 Semantic Service Infrastructures

Since the beginnings of service oriented architectures, the use of explicit semantics for integrating services has been investigated, e.g., by METEOR-S [12]. Recent research has identified best practices and patterns on top of generally used methods, such as REST Web services.

Semantic Automated Discovery and Integration (SADI) [15] defines such design patterns in order to enable semantic REST Web service discovery, messaging, and service description. Among other aspects, it defines that semantically interoperable services should provide their service description on an HTTP GET request and invoke the service with a POST request. The services receive and send instances of OWL classes. The lightweight approach is a good foundation for simple service discovery and interaction, but it is no replacement for a directory service and lacks any support for asynchronous messaging or other features provided by a messaging infrastructure.

The Simple Semantic Web Architecture and Protocol (SSWAP) [4] follows similar principles, but also defines a protocol to be used for discovery. It applies a reasoner to generate explicit statements, e.g., of all classes that apply to the input and output a service advertises. With a shared terminology at transaction time, this protocol also enables semantic brokerage between otherwise incompatible clients and servers.

These approaches are relevant to the problem of a flexible integration of services. However, they do not address the issue of asynchronous messaging.

Furthermore, there are approaches for machine-readable service descriptions for REST APIs, such as MicroWSMO and hRESTS [7]. While these are limited to HTTP-based service interfaces, we might consider following a similar approach, perhaps using WSMO-lite [14], to describe and annotate microservices on an AMQP message bus. For example, instead of using URI templates for addresses we might define the routing keys of a topic exchange to deliver a message to the service.

4.2 Semantic Data Processing

In contrast to the above section, this part focuses on research on processing data which is described semantically, i.e., data that is represented in an RDF-related format.

LinkedPipes ETL[9] (formerly UnifiedViews [6]) is an ETL (extract, transform, load) processing environment for semantic data, built in the COMSODE project. It allows to execute tools for different RDF processing tasks using small wrappers called data processing units (DPU). Tools can be chained together to build workflows, which can be debugged by inspecting the data that is passed through the different components step by step. UnifiedViews is flexible enough to support reasonable processes and relatively easy to extend. However, it lacks support for context or more complex data units (the general interface between the tools is an RDF triple), real-time data, and geospatial aspects.

The GeoKnow project focused on semantic processing tools for geospatial data. The GeoKnow Generator [5] contains various tools along the Linked Data lifecycle that have been developed for processing geospatial data. With the GeoKnow Generator Workbench [3], users can configure and invoke the tools in a Web interface, including the execution and monitoring of long-running tasks as a batch process. In the project, several tools, such as the interlinking framework LIMES [8], have been extended to provide support and optimized processing of geospatial information. While this research provides important fundamentals on which GEISER can build upon, it lacks any support for automated workflows with multiple tools and data streams as input.

[9] `https://etl.linkedpipes.com/`

In addition to these relatively generic projects and frameworks, there are ongoing research efforts relevant to the presented use cases. For instance, the EW-Shopp [10] project targets marketing services which incorporate contextual data, such as weather conditions and events such as holidays. While this is similar to the data-driven geomarketing use case, GEISER focuses on providing respective insights to local businesses.

5 Conclusions

In this paper we described the motivation and architecture of the GEISER platform. In comparison to related work, it will enable the scalable processing of geospatial and sensor data with semantic technologies.

Services for each Linked Data lifecycle phase, as well as lower-level generic infrastructure services and high-level use case specific ones are connected via a message bus. The platform will also provide features for vertical concerns, such as non-functional requirements.

In future work, we plan to implement the use cases in detail using the platform services. We will evaluate the flexibility of the platform interfaces and the scalability of the messaging middleware by benchmarking the system in the HOBBIT platform [9], comparing the currently chosen AMQP implementation against alternatives such as ZeroMQ. Further extensions are planned w.r.t. the components for extracting, storing, interlinking and querying data. For example, to the best of our knowledge, so far there are no interlinking frameworks with explicit support for data streams.

Acknowledgement

The authors would like to thank Michael Schmidt from metaphacts for creating the GEISER architecture figure.

The research and development project that forms the basis for this report is funded under project No. 01MD16014E (GEISER) within the scope of the Smart Services World technology programme run by the Federal Ministry for Economic Affairs and Energy and is managed by the project management agency at the German Aerospace Center (DLR-PT). The authors are responsible for the contents of this publication.

Bibliography

[1] Dobbelaere, P., Esmaili, K.S.: Kafka versus rabbitmq: A comparative study of two industry reference publish/subscribe implementations: Industry paper. In: Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems. pp. 227–238. DEBS '17, ACM, New York, NY, USA (2017), http://doi.acm.org/10.1145/3093742.3093908

[2] Ermilov, I., Ngonga Ngomo, A.C.: Taipan: Automatic property mapping for tabular data. In: Proceedings of 20th International Conference on Knowledge Engineering and Knowledge Management (EKAW'2016) (2016), http://svn.aksw.org/papers/2016/EKAW_Taipan/public.pdf

[3] Garcia-Rojas, A., Hladky, D., Wauer, M., Isele, R., Stadler, C., Lehmann, J.: The geoknow generator workbench: An integration platform for geospatial data. In: Proceedings of the 3rd International Workshop on Semantic Web Enterprise Adoption and Best Practice (2015)

[4] Gessler, D.D., Schiltz, G.S., May, G.D., Avraham, S., Town, C.D., Grant, D., Nelson, R.T.: Sswap: A simple semantic web architecture and protocol for semantic web services. BMC bioinformatics 10(1), 309 (2009)

[5] Grange, J.J.L., Lehmann, J., Athanasiou, S., Rojas, A.G., Giannopoulos, G., Hladky, D., Isele, R., Ngonga Ngomo, A.C., Sherif, M.A., Stadler, C., Wauer, M.: The geoknow generator: Managing geospatial data in the linked data web. In: Proceedings of the Linking Geospatial Data Workshop (2014), http://jens-lehmann.org/files/2014/lgd_geoknow_generator.pdf

[6] Knap, T., Kukhar, M., Machác, B., Skoda, P., Tomes, J., Vojt, J.: Unifiedviews: An etl framework for sustainable rdf data processing. In: Presutti, V., Blomqvist, E., Troncy, R., Sack, H., Papadakis, I., Tordai, A. (eds.) ESWC (Satellite Events). Lecture Notes in Computer Science, vol. 8798, pp. 379–383. Springer (2014)

[7] Maleshkova, M., Pedrinaci, C., Domingue, J.: Supporting the creation of semantic restful service descriptions. In: In workshop: Service Matchmaking and Resource Retrieval in the Semanti Web (SMR2) at 8th International Semantic Web Conference (2009)

[8] Ngonga Ngomo, A.C., Auer, S.: Limes - a time-efficient approach for large-scale link discovery on the web of data. In: Proceedings of IJCAI (2011)

[9] Ngonga Ngomo, A.C., Röder, M.: HOBBIT: Holistic benchmarking for big linked data. In: ESWC, EU networking session (2016), http://svn.aksw.org/papers/2016/ESWC_HOBBIT_EUNetworking/public.pdf

[10] Palmonari, M., Roman, D., Grobelnik, M., Marguglio, A., Paoli, F.D., Nokolov, N., Kosmerlj, A., Gallina, C.: Supporting event and weather-based data analytics and marketing along the shopper journey. In: Extended Semantic Web Conference 2017. Portoroz, Slovenia (May 28-June 1 2017), http://www.ew-shopp.eu/wp-content/uploads/2017/07/EW-Shopp-ESWC2017_Poster.pdf

[11] Sherif, M.A., Dreßler, K., Smeros, P., Ngonga Ngomo, A.C.: RADON - Rapid Discovery of Topological Relations. In: Proceedings of The Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17) (2017), `https://svn.aksw.org/papers/2017/AAAI_RADON/public.pdf`

[12] Sheth, A.P., Gomadam, K., Ranabahu, A.: Semantics enhanced services: Meteor-s, sawsdl and sa-rest. IEEE Data Eng. Bull. 31(3), 8–12 (2008)

[13] Speck, R., Röder, M., Oramas, S., Espinosa-Anke, L., Ngonga Ngomo, A.C.: Open knowledge extraction challenge 2017. In: Semantic Web Evaluation Challenge. pp. 35–48. Springer International Publishing (2017), `https://svn.aksw.org/papers/2017/ESWC_Challenge_OKE/public.pdf`

[14] Vitvar, T., Kopecky, J., Viskova, J., Fensel, D.: Wsmo-lite annotations for web services. In: Hauswirth, M., Koubarakis, M., Bechhofer, S. (eds.) Proceedings of the 5th European Semantic Web Conference. LNCS, Springer Verlag, Berlin, Heidelberg (June 2008), `http://data.semanticweb.org/conference/eswc/2008/papers/281`

[15] Wilkinson, M.D., Vandervalk, B., McCarthy, L.: The semantic automated discovery and integration (sadi) web service design-pattern, api and reference implementation. Journal of biomedical semantics 2(1), 8 (2011)