Taming the Evolution of Big Data and its Technologies in BigGIS
A Conceptual Architectural Framework for Spatio-Temporal Analytics at Scale

Patrick Wiener[1], Viliam Simko[2] and Jens Nimis[1]
[1]*Karlsruhe University of Applied Sciences, Karlsruhe, Germany*
[2]*FZI Research Center for Information Technology, Karlsruhe, Germany*

Keywords: Big Data, Geographic Information Systems, Software Architectures, Software Design Patterns, Data Pipelines.

Abstract: In the era of spatio-temporal big data, geographic information systems have to deal with a myriad of big data induced challenges such as scalability, flexibility or fault-tolerance. Furthermore, the rapid evolution of the underlying, occasionally competing big data ecosystems inevitably needs to be taken into account from the early system design phase. In order to generate valuable knowledge from spatio-temporal big data, a holistic approach manifested in an appropriate architectural design is necessary, which is a non-trivial task with regards to the tremendous design space. Therefore, we present the conceptual architectural framework of *BigGIS*, a predictive and prescriptive spatio-temporal analytics platform, that integrates big data analytics, semantic web technologies and visual analytics methodologies in our continuous refinement model.

1 INTRODUCTION

Geographic information systems (GIS) have long been used to support humans in complex decision-making processes (Crossland et al., 1995) such as transport logistics, environment protection or civil planning. They are supported by an ever-growing variety and volume of new data sources such as hyperspectral imagery from unmanned aerial vehicles, real-time sensor data streams, or open geodata initiatives and are often expected to deliver their analysis results in a timely or even interactive fashion (OGC, 2013). As the described properties are the defining cornerstones in the field of big data, it is inevitable that the respective methodologies and technologies become integral part of future GIS (Peng and Liangcun, 2014). Such big data enabled GIS have to provide core functionalities for spatio-temporal analytics with all the required subtasks such as an integrated treatment of raster and vector data. Moreover, as big data itself provides a rapidly developing ecosystems of tools and infrastructures, future GIS will need to cope with heterogeneity not only on data and use case but also on infrastructure level. The manifold roles of humans in GIS decision-making processes as users, experts and sometimes even as data providers, have to be reflected in GIS by providing appropriate interaction capabilities and resilience to uncertainty. An application that fulfills all these different and complex requirements may need to perform a variety of tasks on the information that it processes. A straightforward but inflexible approach to implementing such applications would be to perform this processing in one monolithic module. However, this approach is likely to reduce the opportunities for refactoring code, optimizing it, or reusing it if similar processing (sub-)tasks are required elsewhere within the application or in other scenarios. Thus, decomposing complex processing tasks into a series of discrete and reusable components is a central idea of our BigGIS conceptual architectural framework. The utilization of the established pipes and filters pattern (Buschmann et al., 2007) to construct the overall application can improve performance, eases scalability and reusability by allowing components that perform processing and analysis tasks to be deployed and scaled independently.

In summary, in this paper we pursue the goal to provide a highly-flexible, modular, scalable and fault-tolerant architectural framework for a wide range of batch and streaming workloads to process and analyze heterogeneous and uncertain spatio-temporal data at scale by leveraging existing big data technologies. Thereby, we instantiate the continuous refinement model of our BigGIS vision (Wiener et al., 2016).

To achieve the above goal, we first discuss related work in Section 2. In Section 3, we describe four exemplary BigGIS applications and therefrom derive the architectural framework's requirements. Section 4 is

Wiener, P., Simko, V. and Nimis, J.
Taming the Evolution of Big Data and its Technologies in BigGIS - A Conceptual Architectural Framework for Spatio-Temporal Analytics at Scale.
DOI: 10.5220/0006334200900101
In *Proceedings of the 3rd International Conference on Geographical Information Systems Theory, Applications and Management (GISTAM 2017)*, pages 90-101
ISBN: 978-989-758-252-3

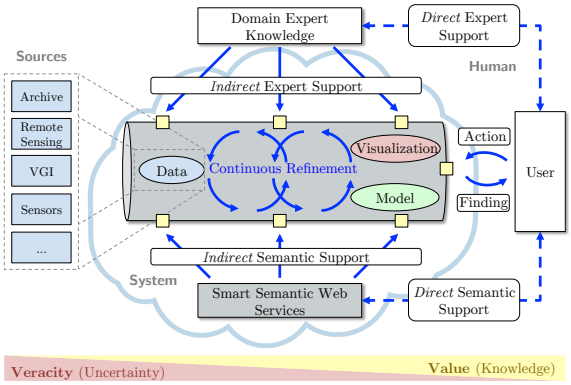

Figure 1: Continuous refinement model in BigGIS (Wiener et al., 2016).

dedicated to the architectural elements of the framework while Section 5 describes how they work together along a user's system interaction stages. To discuss and illustrate our approach we show an example implementation for one of the use cases in Section 6. We conclude the paper and give an outlook on future work in the last section.

2 RELATED WORK

In our previous work, we have introduced the vision of *BigGIS*, a next generation geographic information systems shown in Figure 1 that allows for predictive and prescriptive spatio-temporal analytics of geographic big data (Wiener et al., 2016). We consider uncertainty to be reciprocally related to generating new insights and consequently knowledge. Thus, modeling uncertainty is a crucial task. On an abstract level, our approach extends the knowledge generation model for visual analytics (Sacha et al., 2014) in an integrated analytics pipeline which blends big data analytics and semantic web technologies on system-side with domain expert knowledge on human-side, thereby allowing expert and semantic knowledge to enter the pipeline at arbitrary stages what we refer to as refinement gates. By leveraging the continuous refinement model, we present a holistic approach that explicitly deals with all big data dimensions. By integrating the user in the process, computers can learn from the cognitive and perceptive skills of human analysis to create hidden connections between data and the problem domain. This helps to decrease the noise and uncertainty and allows to build up trust in the analysis results on user side which will eventually lead to an increasing likelihood of relevant findings and generated knowledge.

The conversion of our vision into a more concrete architectural framework is at its heart a system design issue and as such should benefit from and rely on the extensive experience in this field which is externalized in general software architecture patterns (Buschmann et al., 2007). E.g. *layering* and *pipes and filters* are two often occurring and re-used architecture patterns where the latter also is the foundation of the BigGIS architecture framework. The BigGIS framework design decision to build up on the pipes and filters pattern is in line with prominent representatives of general big data architecture such as the *lambda architecture* (Marz and Warren, 2013) or *kappa architecture* (Kreps, 2014). They both are based on the pipes and filters pattern while trying to cope with the tension between batch and stream processing in big data analytics.

There exists a number of big data systems and platforms that follow a pipes and filters related approach. *StreamPipes* (Riemer et al., 2015) provides a user-oriented interface for managing complex event processing on top of big data streams. It leverages semantic web technologies to describe elements of the pipelines which are then running on a variety of distributed big data platforms. Reliable and scalable messaging across the various components and technologies is achieved through Apache Kafka[1]. The *KNIME Analytics Platform* (Berthold et al., 2007) provides a GUI for building data processing pipelines that can run locally or in a KNIME cluster. A so-called KNIME workflow is composed of nodes connected by edges between their input/output ports. When executed, pipelines operate in batch manner, exchanging data tables, predictive models, parameters or connections to external services. The KNIME platform provides generic as well as domain-specific nodes for, e.g., chromosome analysis, machine learning, time series analytics. The purpose of *Apache NiFi* (Apache Foundation, 2016) is to allow for high-performance data flow management throughout an enterprise. Therefore, it provides a user-oriented graphical interface utilizing powerful and scalable directed graphs to capture data routing, transformation and system mediation logic. However, at the time of writing, these platforms do not provide spatial analytics capabilities for the wide spectrum of GIS use cases in order to address all big data induced requirements.

Even more specific within the spatio-temporal analytics domain a number of systems and libraries have originated, e.g. *PlanetSense* (Thakur et al., 2015), *ArcGIS Big Data Analytics* (Esri, 2016), *GeoTrellis* (Eclipse Foundation, 2016), and many more. While they all provide certain functionality within the scope of BigGIS' applications, with their respective alignment they only address a subset of the use cases and derived requirements presented in the following.

[1] https://kafka.apache.org/

3 MOTIVATING USE CASES

In this section, we present four motivating use cases which demonstrate different aspects of geospatial and spatio-temporal analytics that we want to support in BigGIS in a scalable way using the state-of-the-art technologies from the big data domain. An overview of all use cases is summarized in Table 1 and the resulting requirements are discussed in Subsection 3.5.

3.1 Hot Spot Analysis on New York Taxi Drop-offs

This use case is motivated by the ACM SIGSPATIAL GISCUP 2016 competition which aims at finding the top 50 hot spots in space and time in terms of taxi drop-off locations and passenger counts by computing the Getis-Ord G* statistics (Ord and Getis, 1995). The dataset contains taxi drop-offs from New York, years 2009-2015. Computing the G* statistics in a grid involves aggregation of points into grid cells, computing the convolution, sorting and optionally computing the mean and standard deviation of the whole dataset. A graphical overview of the pipeline is depicted in Figure 2.

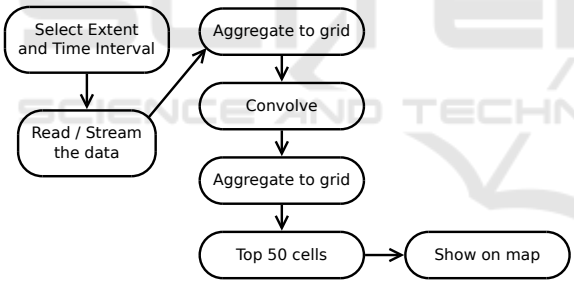

Figure 2: Hot spot analysis pipeline based on Getis-Ord G* statistics.

Files are stored in HDFS[2] (Hadoop Distributed File System) using CSV (comma separated values) format; 2GB per month of data. Every row represents a single taxi drop-off whereas the columns show corresponding features such as latitude, longitude, time and passenger count. The data points have to be aggregated into a spatio-temporal grid with a granularity of approx. $100\ m \times 100\ m \times 1\ day$. The grid cells are then convoluted with their neighbour cells (queen-distance of one cell, i.e., 27 cells in a space-time cube) and the top 50 cells are then selected.

If the actual z-scores need to be computed, the grid-based G* algorithm is used (Def. 1) which requires the mean and standard deviation of the whole dataset.

[2]http://hadoop.apache.org/

Def. 1 (G^* in a grid)**.** *Assuming a notation $X \overset{\mathtt{op}}{\circ} W$ to denote a focal operation op applied on an n-dimensional grid X with a focal window determined by an n-dimensional matrix W. The function G^* can be expressed as follows:*

$$G^*(X,W,N,M,S) = \frac{X \overset{\mathtt{sum}}{\circ} W - M.\sum_{w \in W} w}{S\sqrt{\frac{N.\sum_{w \in W} w^2 - (\sum_{w \in W} w)^2}{N-1}}}$$

where:

- *X is the input grid.*
- *W is a weight matrix of values between 0 and 1.*
- *N represents the number of all cells in X.*
- *M represents the global mean of X.*
- *S represents the global standard deviation of X.*

In the competition, Apache Spark[3] is used for computing the space-time cubes of the datasets restricted to year 2015 and to an envelope encompassing the five New York City boroughs.

This is a typical batch processing use case which can be extended to a stream processing application, e.g, by updating the top 50 hot spots on-the-fly. There are multiple parameters that can be configured by the users including cell size, the number of top-k hot spots returned or the spatio-temporal extent of the analysis.

3.2 Computing NDVI/NDWI from Landsat Images

This is a raster processing use case that involves local map algebra operations (Tomlin, 1990) applied on a multiband raster. We use the GeoTrellis (Eclipse Foundation, 2016) library for distributed raster processsing which splits an input raster into uniform tiles indexed using a space filling curve. The use case is loosely inspired by the *GeoTrellis landsat tutorial project* (Emanuele, 2016). It involves the following steps: (1) Discovery of landsat images. (2) Downloading the discovered GeoTIFF rasters from Amazon S3 cloud. (3) Creating a 3-band GeoTIFF from the red, green and nir (near infrared) bands masked with the quality assessment (QA) band. (4) Computing normalized differenced vegetation index (NDVI) and normalized difference water index (NDWI) from the red, green and near infrared bands. (5) Reprojecting the GeoTIFF into WGS-84[4] projection (6) Tiling and indexing of the raster into a GeoTrellis catalog (using a space-filling curve) (7) Building a tile pyramid for different zoom levels that can be served to web clients in an efficient manner given $x/y/z$ coordinates. An overview of the pipeline is shown in Figure 3.

[3]https://spark.apache.org/

[4]WGS 84 / Pseudo-Mercator, http://epsg.io/3857

Table 1: Summary of use cases presented as a motivation. **Batch** processing: Data processed once on request till the end of the dataset; **Stream** processing: Data continously processed from a potentially infinite stream; **Time** dimension: The data are located in space and time. **Raster** processing: Usually GeoTIFFs, including map algebra operations; **Vector** processing: Usually point measurements; **Model**: Uses some prediciton model such as SVM, Logistic regression, etc. Involves training, prediction and evaluation of the model.

	Batch	Stream	Time	Raster	Vector	Model
Hot Spot Analysis on New York Taxi Drop-offs	✓	(✓)	✓	×	✓	×
Computing NDVI/NDWI from Landsat Images	✓	(✓)	×	✓	×	×
Stream Enrichment	×	✓	✓	✓	✓	×
Land Use Classification & Change Detection	✓	✓	✓	✓	✓	✓

Legend: ✓ Yes, × No, (✓) Can be extended

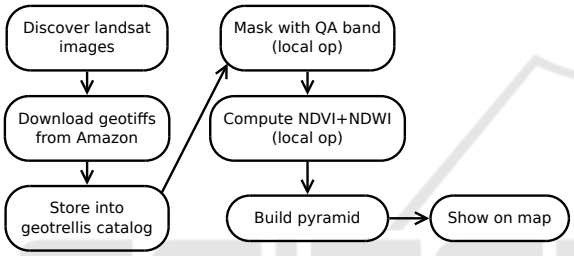

Figure 3: NDVI/NDWI computation pipeline for Landsat images.

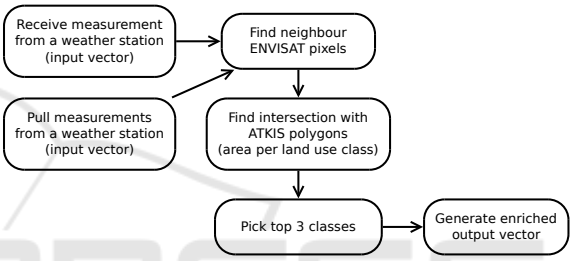

Figure 4: Overview of the stream enrichment pipeline.

3.3 Stream Enrichment

An overview of this scenario is depicted in Figure 4. The goal is to treat historical and newly measured temperatures from weather stations[5] as a stream of *input vectors* (Def. 2) that has to be enriched with additional information from raster and vector layers – *output vectors* (Def. 3). Besides the air temperatures, the stream may contain additional features, such as humidity or wind speed/direction.

Def. 2 (Input vector)**.** *A single measurement from weather station id is encoded as a vector:*

$$IN_n = (id, ts, lat, lon, temp, hum, \ldots, wind)$$

- *ts is the timestamp of the measurement*
- lat, lon *are geographical coordinates in WGS-84*
- $temp, hum, \ldots, wind$ *are measured values such as air temperature, humidity, wind speed etc.*

Envisat rasters represent hourly LST (Land Surface Temperature) readings with pixels of size 0.05×0.05 *deg* (i.e. 5.6×3.7 *km* in Baden-Württemberg). In the analysis, the nearest pixels were needed for the analysis. Although the satellites provide hourly scans of the same region, gaps might have occurred due to bad weather conditions or other factors, resulting in NA values. The vector layer is the LUBW ATKIS database that contains land use classification polygons for regions such as forest, urban area, railway etc.

Def. 3 (Enriched output vector)**.** *For a single measurement* $IN_n = (id, ts, lat, lon, temp, hum, \ldots, wind)$ *as defined in Def. 2, we construct an output vector* OUT_n *as follows:*

$$\begin{aligned} OUT_n \quad = \quad & (id, ts, lat, lon, temp, hum, \ldots, wind, \\ & pdist_1, lst_1, c_1^{\#1}, a_1^{\#1}, c_1^{\#2}, a_1^{\#2}, c_1^{\#3}, a_1^{\#3}, \\ & \ldots, \\ & pdist_9, lst_9, c_9^{\#1}, a_9^{\#1}, c_9^{\#2}, a_9^{\#2}, c_9^{\#3}, a_9^{\#3}) \end{aligned}$$

There are potentially nine raster pixels $i \in \{1, \ldots, 9\}$ *that are nearest to the location* (ts, lat, lon)*. For a given pixel i, let* lst_i *be its land surface temperature and* $pdist_i$ *be the distance from the pixel center to the location* (lat, lon)*. Let* $c_i^{\#j}$ *be the land use class of the ATKIS polygon that has the j-th largest area of intersection* $a_i^{\#j}$ *with the pixel i.*

The stream of output vectors is passed through a

[5]For evaluation purposes, we use weather stations from Deutscher Wetterdienst, LUBW Landesanstalt für Umwelt, Messungen und Naturschutz Baden-Württemberg and our own LoRA-based sensors.

dedicated queue, e.g. in Apache Kafka, and can be further processed in a distributed manner.

3.4 Land Use Classification and Change Detection

This scenario involves distributed raster processing combined with machine learning. The goal is to automatically detect changes in the land use classification based on aerial/satellite imagery. The challenge here is that *land use classes*, indicating how people are using the land, have a higher semantic meaning and thus cannot be directly determined from the images. On the other hand, *land cover classes*, indicating the physical land type (e.g. green area, water etc.), can be determined by analyzing the images in an automated way. Land use classification databases are updated manually by domain experts based on land cover surveys from multiple years with the help of predefined compatibility criteria between land use and land cover classes (e.g. how much coverage of class A may be contained in a land use class B). Some of these criteria can be checked automatically thus minimizing the amount of work for domain experts. An overview of the whole use case is depicted in Figure 5.

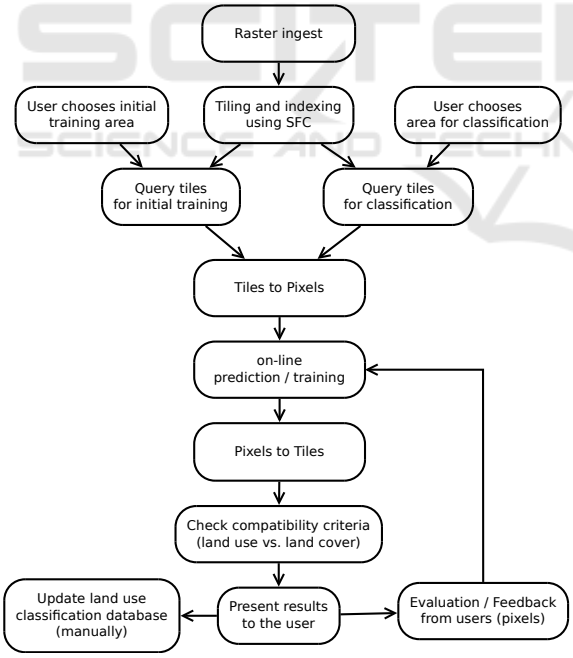

Figure 5: Land use classification and change detection pipeline.

Land use classification is encoded as polygons with an attached class. The aerial and satellite images are encoded as multiband rasters. In order to support distributed processing, polygons have to be indexed using a space-filling curve (SFC). The rasters have to be cut into uniform tiles (e.g. 256×256 *px*), organized in a grid and also indexed using a SFC for efficient distributed processing. An important architectural feature, besides the local and focal map algebra operations (Tomlin, 1990), is the ability to convert raster tiles into a stream of individual pixels (Def. 4).

Def. 4 (Converting Tiles to a Stream of Pixels)**.** *Let T be a square multiband tile with $n+1$ bands.*

$$T = (T^{idx}, T^{v_1}, \ldots, T^{v_n})$$

T^{idx} is a composed index containing the SFC index of the tile and row/column pixel coordinates within the tile. $T^{v_1}, \ldots, T^{v_n}$ represent the pixel values, (e.g. red, green, blue, nir). Then, we define a mapping function from tile to pixels as follows:

$$\psi : T \mapsto \{T_{i,j} : \forall i, j\}$$

Notice that result of $\psi(T)$ is a sample set, where each element is a tuple $(idx, v_1, \ldots, v_n)$.

We can also define a corresponding inverse function ψ^{-1} that converts stream of pixels back to a multiband raster.

The pixelization function ψ preprocesses the raster tiles into a form suitable for pixel classification. Here, we can either follow the *traditional approach* or the *online stream learning approach*. In the former case, the dataset is split into training and testing set (manually or automatically). The training set is used for building the classification model whose prediction performance is crossvalidated on the testing set. In the latter, the prediction performance of a model is continuously evaluated and adjusted (if supported by the model) or retrained after reaching a certain error threshold. In both cases, the classifier converts incoming pixels into a new stream of predictions. These are converted back to raster tiles using the aforementioned inverse function ψ^{-1}.

After the classification phase, the compatibility criteria between the land use classes and newly predicted land cover classes can be checked and presented to the domain expert for manual inspection. Feedback from the user can be treated as a source of training samples to improve classification accuracy forming a feedback loop.

3.5 Requirements

As the aforementioned use cases demonstrate, the range of application is manifold. Thus, defining distinct functional and non-functional requirements is not feasible, which is why we refer to the more general term of requirements in the following.

3.5.1 Spatial and Temporal Analytics Support

Regarding the domain of spatio-temporal big data analytics, it is mandatory that a big data enabled GIS must provide support for spatio-temporal analytics. For instance, this can be map algebra operations on raster data, vector/raster conversions, spatial time series analysis, or multilayer multiband capabilities.

3.5.2 Heterogeneity-Awareness

Considering the rapidly growing amount of data sources especially in the space-time context, e.g. hyperspectral imagery from unmanned aerial vehicles, it is necessary to provide means to (1) easily integrate these heterogeneous data sources through defined wrappers and transformations in a required and more standardized format before analyses, as well as (2) extend the pool of possible data sources to new ones.

3.5.3 Uncertainty-Awareness

Since noise and erroneous data are natural in the real world, additional provenance and metadata information, e.g., the type and accuracy of a given sensor, can be beneficial for modeling these inherent uncertainties in data. Thus, the system should allow annotating the semantic information and relation of data sources.

3.5.4 User Integration

While computers are good at data management and processing, the humans' cognitive and perceptive skills allow to establish hidden connections between the data and the problem domain. Thus, providing a set of suitable web-based visualizations is obligatory, to (1) present analyzed results to users in an adequate manner, (2) allow expert users to visually explore the results, (3) provide a channel to manipulate computation at arbitrary stages in the processing pipeline.

3.5.5 Technology-Agnosticism

The ever ongoing evolution in computer science in terms of advancements in big data analytics, for instance enhanced distributed machine learning algorithms and newly arising open source big data frameworks, introduce crucial requirements that dictate our design approach. Choosing one technology in favor of another can therefore cause strong limitations in terms of applicability to a wider range of use cases, thus introducing lock-in effects and inflexibility. As a consequence, the architectural design should account for these circumstances such that it is possible to leverage multiple underlying big data technologies for distributed batch, stream processing or machine learning as needed. By splitting up computational monoliths in a variety of discrete blocks, each serving a certain functionality and potentially running on different big data technology, invokes the requirement of having a standardized and reliable way of communication as well as repositories to store these artifacts for later use. Therefore, intermediate results of these computational blocks should be made available for arbitrary consecutive ones such that it is possible to apply different functions on the same data in parallel.

4 ARCHITECTURAL ELEMENTS

Designing a conceptual architectural framework, which accounts for the aforementioned requirements, involves decomposing the design in various architectural elements that are presented in the following.

Def. 5 (BigGIS)**.** *The conceptual architectural framework of BigGIS is formally described as a tuple* (P,S,R,A)*, where:*

- P *is a set of pipelines*
- S *is a set of services*
- R *is a set of repositories*
- A *is a set of user actions*

4.1 Pipelines

In BigGIS, functionalities are encapsulated in discrete and reusable computational blocks called *nodes* each of which follow the single responsibility principle and perform only one specific task, e.g., download source data, perform spatial binning, or apply a classifier to predict the land use class. Multiple consecutive nodes are interconnected through dedicated *queues* to transfer data in a standardized and homogeneous format between nodes. The resulting processing chain allows for a combination of these nodes into a use case specific *pipeline* as is generically shown in Figure 6.

Def. 6 (Pipeline)**.** *A pipeline p is formally described as a tuple* (N,Q)*, where:*

- N *is a set of nodes*
- Q *is a set of queues*

4.1.1 Nodes

A node n is responsible for processing and analyzing input data that it consumes from one or more input queues Q_{in} by subscribing to dedicated topics T. In-

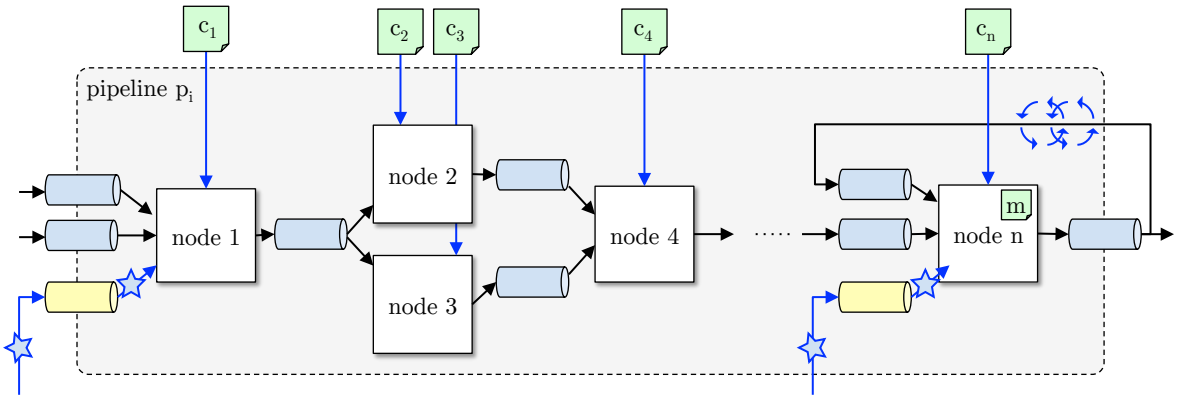

Figure 6: A pipeline in BigGIS is composed of consecutive nodes and queues, in which each node i performs a dedicated processing task, e.g. normalizing, cleansing, filtering, or analytical task, e.g. calculating statistics, applying pre-trained machine learning models m, on the received input data (input queues). Generally, the results are propagated downstream (output queues) for further processing but can also be back propagated for iterative algorithms. Configuration parameters c_i describe the setup a node. Special input queues called refinement gates allow for external user knowledge to update/change the corresponding node in the form of periodical refinement events (★).

side a node, task specific functions F are consecutively applied whenever new data D is received on the input side to further manipulate and refine the input data. In addition, configuration parameters c from the config repository are used to configure and parametrize the node. The results of the computation is then published back to one or more output queues Q_{out} on new topics T. By constraining the semantics of the input data queue mechanism, we want to solve the following issues: (1) preventing deadlocks, (2) maximizing performance.

While it can be necessary for certain nodes to join multiple input queues together, this simplification avoids deadlocks and idling nodes which increases the performance. Besides that, consecutive nodes need not to be deployed on the same computational framework. Thus, a node can run on the most suitable underlying big data framework and in a distributed manner. To continuously refine results nodes can subscribe to their own output queue and consume the results D^*, e.g. to perform iterative computation. An essential property of a node is its metadata *meta* which provides information characterizing the node, for instance what the node's job is, or what type of input data and formats it can handle. A special type of node is the machine learning node where a suitable pre-trained machine learning model m of set M is loaded inside the node as shown in Figure 7, to perform predictive analysis, e.g. to classify the land use class as a crucial part to automated change detection as depicted in Section 3.4.

Def. 7. *(Node). A node n is formally described as a tuple* $(Q_{in}, Q_{out}, F, c, meta, m)$*, where:*

- Q_{in} *is a set of input queues*
- Q_{out} *is a set of output queues*
- F *is a set of functions*
- c *are node specific configurations and parameters*
- *meta is the metadata description*
- m *is a machine learning model of set* M *(optional)*

4.1.2 Queues

A queue q is a communication channel either between (1) external and internal components, e.g. from external data sources and nodes, (2) two or more consecutive nodes, (3) nodes and data sinks, e.g. a database or visual analytics user interfaces. Queues are an integral part of BigGIS as they provide a flexible way to decouple arbitrary components such as two or more consecutive nodes. A queue carries data d associated with a certain topic t that has been published by specific node. Arbitrary number of subsequent nodes can then register for this topic in order to dequeue the data in a first-in-first-out manner. Queues can either be classified as input queues Q_{in} or as output queues Q_{out} with respect to a certain node. The data can be the actual data, e.g., a stream of vector data from weather stations, or metadata information, e.g., storage location of raster images. Besides, there are special input queues called refinement gates Q_{rgate} that contain periodical refinement events (★) representing external user knowledge that will trigger a certain update in the corresponding node, e.g. specifying the number of top-k hot spots to be returned by a top-k hot spots node, or providing new training data samples for a machine learning node, as shown in Figure 7.

Def. 8. *(Queue). A queue q is formally described as a tuple* (T, D)*, where:*

- T *is a set of topics*
- D *is the available data*

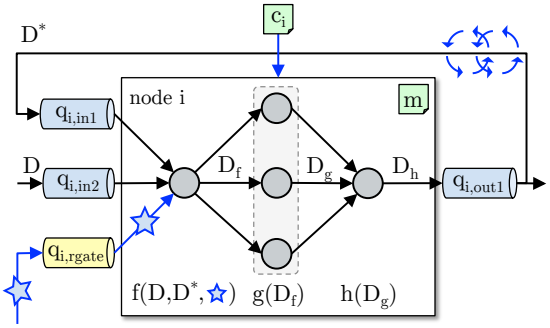

Figure 7: An iterative machine learning node i making predictions on data D from corresponding input queue $q_{i,in}$ by applying the functions f, g, h and publishing the results on the output queue $q_{i,out}$, while in this case the predicted output D^* is self-subscribed. User knowledge, e.g. for parameter tuning, gets injected through the refinement gate $q_{i,rgate}$.

4.2 Services

Self-sustained units of functionalities for a given task that expose a distinct interface for interaction are called *services S*. In BigGIS, a service can be classified in three categories that are (1) provider, (2) integrator, (3) manager. Providers expose adapters to connect to certain data sources and data sinks. Reasoning on the semantic metadata as part of the data integration process is the task of the integrators, which are responsible for data homogenization and normalization. Managers take care of the deployment and supervision of pipelines and nodes as well as handling user actions such as updating configurations and parameters, or injecting new refinement events to refinement gates.

Def. 9. *(Service). A service s is formally described as a tuple* (s_t, s_n)*, where:*

- s_t *is the service type*
- s_n *is the service name*

4.2.1 Data Source Service

The data source service provides a set of adapters to external data sources, e.g. satellite images stored in Amazon S3 buckets, volunteered geographic information and other open data initiatives through REST API calls, or streaming sources such as sensor readings from weather stations.

4.2.2 Data Sink Service

Like the aforementioned data source service, the data sink service provides a set of adapters to expose the processed and analyzed data from the last output queues of a pipeline to a variety of different data sinks, e.g., databases for persistent storage, REST APIs for exposing the top-k hot spots, or visual representations such as real-time monitoring dashboards or specific visual analytic views. The latter presents a fundamental element in BigGIS that enables users to visually explore the data and analyses results thereby allowing them (1) to reason on and interpret the massive amounts of spatio-temporal big data to gain insights into causalities of determining factors for a given problem domain that would otherwise not be easily identified, as well as (2) to adjust specific parameters in deployed nodes of pipelines.

4.2.3 Linked API Service

The linked API service is a semantic web service that helps integrating various data sources (integrator), which are typically exposing diverse data formats and data schemas. In collaboration with the data source service and the data source repository, it is possible to perform a smart data integration by building on top of exisiting and well-established ontologies, e.g. QUDT[6], or GeoSPARQL [7] in order to cope with heterogeneous data sources. This allows for (1) traceability and provenance information of spatio-temporal big data, (2) a robust data integration, possibly even of open linked geo data sources, (3) a flexible approach to cope with the heterogeneity in data input and output formats.

4.2.4 Cognitive App Service

The cognitive app service enables pipeline and node deployment as well as user action mediation between visual analytics views, nodes and the underlying big data frameworks (manager). When necessary, this service updates configuration parameters of nodes in the configuration repository and propagates refinement events to update functions of dedicated nodes in the pipeline through refinement gates input queues. In addition, the cognitive app service is context-aware which means that besides the actual deployment of nodes and pipelines it further supervises the execution of the nodes on the underlying big data frameworks.

4.3 Repositories

Dedicated stores for various components in BigGIS are called *repositories R*. Repositories serve two specific purposes (1) conduct metadata information such as information about semantically described data sources, e.g. resolution and update cycles of satellite imagery or pipeline descriptions for performing complex analytical tasks, e.g. land use classification, (2) serve as a storage location from which software artifacts such as

[6] http://www.qudt.org/
[7] http://www.opengeospatial.org/standards/geosparql

packaged nodes or visualizations may be retrieved and deployed on the underlying big data infrastructure.

Def. 10. *(Repository). A repository r is formally described as a tuple* (r_t, r_n)*, where:*

- r_t *is the repository type, either metadata or artifact storage*
- r_n *is the repository name*

4.3.1 Data Source Repository

The data source repository contains semantically annotated descriptions of various data sources that model complex correlations in a graphical approach based on RDF[8] (resource description framework).

4.3.2 Model Repository

The model repository stores pre-trained machine learning models in a reusable way, e.g., PMML (Guazzelli et al., 2009), so that they can be loaded inside a dedicated machine learning node for performing predictive analyses on new data.

4.3.3 Configuration Repository

The configuration repository contains relevant configuration files and parameters for running nodes.

4.3.4 Pipeline Repository

The pipeline repository stores templates and descriptions for performing recurrent and complex analytical tasks. A description contains various useful information, e.g. what data sources and nodes are used, how they are plugged together, or what the parameter values are in order to get a satisfactory prediction quality. This can be used as a good starting point for non-expert users as well as for expert users who can focus on the interpretation and reasoning rather than building a certain pipeline over and over again.

4.3.5 Node Repository

The node repository is an artifact store that comprises all nodes. The artifact itself contains the pre-packaged source code and necessary libraries to perform the task as well as a metadata description of the node, e.g., what the node's task is, what type of input data it can handle, what type of output data it produces, or what underlying big data framework it utilizes.

[8]https://www.w3.org/RDF/

4.3.6 Visualization Repository

The visualization repository is another artifact store that contains predefined web-based graphical user interfaces. On one hand, this could be a configurable real-time dashboard to monitor the analysis results. On the other hand, this could be interactive visual analytics views that enable users to interact with dedicated nodes inside pipelines while visually exploring the data, e.g., changing the time window for computing the mean over the resulting finite set of data elements or adjusting parameters in a machine learning node.

4.4 User Actions

User actions A are a crucial part of BigGIS since they enable the user to interact with the system. Generally, there are various types of actions possible. Thus, we differentiate between (1) non-interactive user actions, e.g., selecting relevant data sources or defining pipelines of nodes, and (2) interactive user actions that are characterized by visualizations to support the knowledge generation on user side. The latter gives the user the opportunity to visually explore the results, reason on them and trigger adequate changes according to the level of domain knowledge. User actions on visualization elements, e.g., moving a slider to change top-k hot spots or calculating spatio-temporal statistics for a given spatial extent, generate tangible, unique responses from a visual analytics system (Sacha et al., 2014) such as BigGIS. By applying adjustments on the visual analytics views the user implicitly manipulates the affected processing nodes N at predefined stages of a given pipeline by injecting refinement events over a set of topics T_{rgate} for refinement gate input queues. Thus, enhancing the creativity and curiosity of the user during the course of exploration. From now on, we refer to user actions as interactive user actions.

Def. 11. *(User Action). A user action a is formally described as a tuple* (a_t, N, T_{rgate}, c)*, where:*

- a_t *is user action type*
- N *is a set of nodes that are affected by changes*
- T_{rgate} *is a set of topics for refinement gate queues*
- c *are node specific configurations and parameters*

5 CONCEPTUAL ARCHITECTURAL FRAMEWORK OF BigGIS

This section introduces the conceptual architectural framework of BigGIS and how the previously pre-

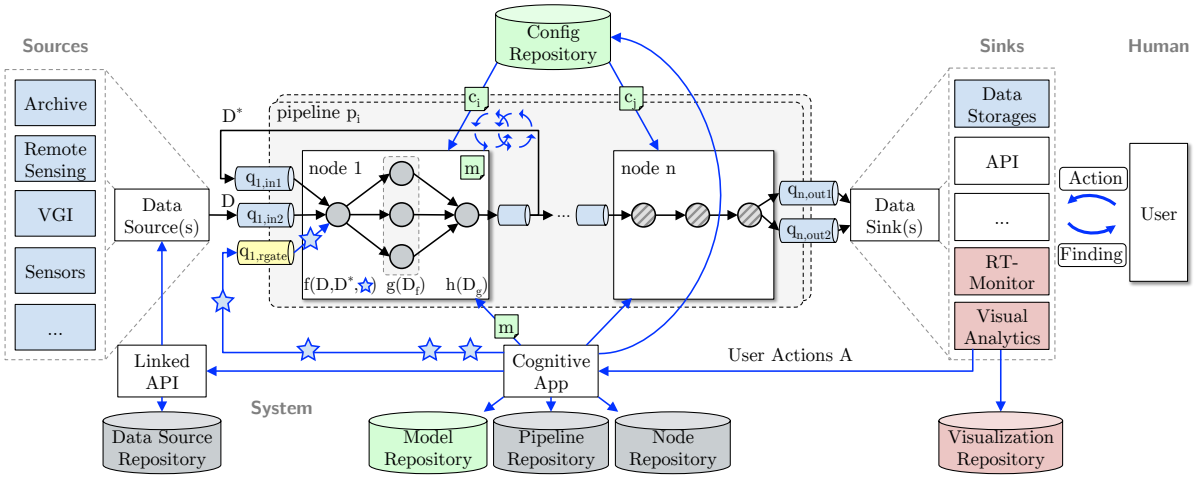

Figure 8: Conceptual Architectural Framework of BigGIS for performing spatio-temporal analytics at scale instantiating the continuous refinement model (Wiener et al., 2016) during the exploration stage. Of particular note is that data flows (→), moving from data sources to data sinks, are countercyclical to information/control flows (←) that originate from the user.

sented architectural elements (Def. 5 – pipelines, repositories, services, user actions) interconnect and thus instantiate the continuous refinement model (Wiener et al., 2016) as shown in Figure 8. This, along with the aforementioned requirements, form the basis of the design considerations. A user interacting with BigGIS would traverse through various stages during her analyses. These stages are (1) preparation, (2) deployment, and (3) exploration, that are discussed in the following.

5.1 Preparation

During the preparation stage, the user prepares for data processing and analyses and either composes a pipeline itself with the necessary nodes and configurations from the node repository or utilizes a predefined pipeline template from the pipeline repository for a specific use case. While the former provides more flexibility the latter further enables a set of suitable visual analytics views from the visualization repository as a designated data sink that allows for manipulation of the processing nodes through refinement gate input queues. Additional configuration of the node can be made by the user in order to suit the needs. Configurations are saved in the configuration repository by the cognitive app service. Generally, the preparation stage consists of data source and data sink selection as well as pipeline composition and configuration.

5.2 Deployment

Once the configuration of the pipeline is completed and the data sources and data sinks are specified, a user action triggers the cognitive app service to deploy the nodes on the underlying big data frameworks. Depending on wether the user has chosen a pipeline template, the cognitive app service looks up the composition in the pipeline repository, loads the appropriate nodes and models from the corresponding repositories and deploys them. Furthermore, the cognitive app service triggers the linked API service to start the data integration process thereby leveraging the semantic annotation and the rules of conversion from the data source repository. The output of the data integration process is published to specified queues from where it gets consumed, continuously analyzed, refined and further distributed to the selected data sinks. In this respect, one eminent data sink type represents the aforementioned pipeline dependent visual analytics views. This way, the results are presented in an adequate way that enhances to cognitive and perceptive skills on user-side to increase the likelihood of relevant findings during the course of exploration.

5.3 Exploration

The exploration stage is characterized by two recurrent steps, that involve (1) reasoning on interesting observations made by the user on retrieved results (finding), e.g. missing data points, patterns in visual representations, or conspicuous predictions of machine learning nodes and (2) manipulating the data processing or models (action). During the course of exploration, the user constantly interacts with the visual analytics views to understand the observations through certain elements, e.g. adjusting parameters for a data heatmap layer on top of OpenStreetMap data. This interactive user action provokes the cognitive app service to update the configurations and parameters in the configuration repository and propagates refinement events (★) over

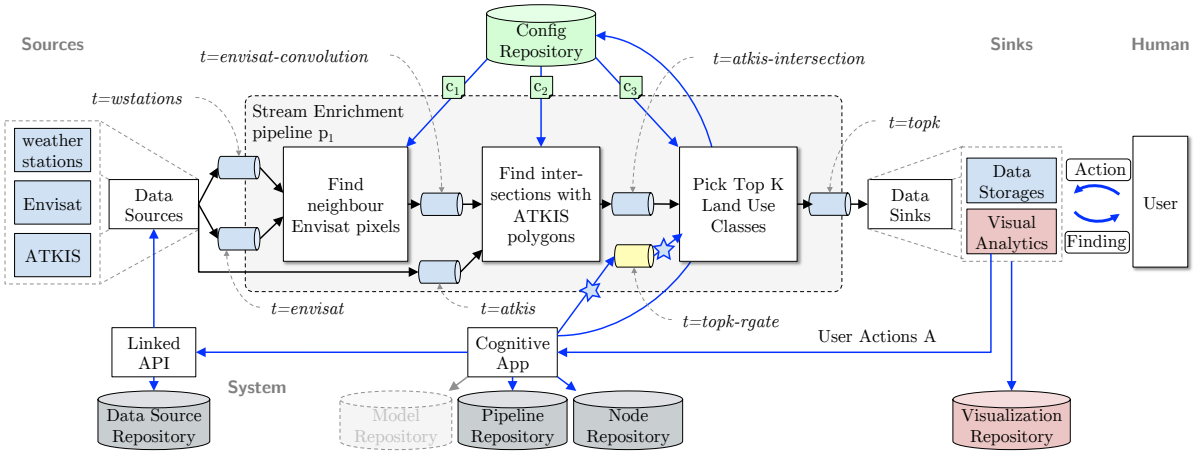

Figure 9: Stream Enrichment use case in BigGIS during the exploration stage showing three nodes and corresponding queue topics *t*, as well as the relevant data sources and data sinks.

the refinement gate input queues to the designated node instances. The node then updates its configuration and parameters to satisfy the adjustments in order for a fast retrieval of newly calculated results to allow to continue reasoning on user-side. The continuous refinement approach is especially beneficial in terms of deployed machine learning nodes, where model parameters can be easily refined to improve the accuracy of the prediction through (1) an iterative approach by subscribing to the own output queue to automatically supervise the process and autotune the parameters, as well as (2) direct user involvement triggered by a refinement event. The latter can involve feedback in types of corrections of predicted labels or parameters itself. Eventually, new insights are generated when the user is able to understand and interpret the findings in the context of the problem domain.

6 DISCUSSION BY EXAMPLE

To discuss our framework, we formally show how the *Stream Enrichment* use case from Section 3.3 is instantiated within BigGIS as depicted in Figure 9, which is presenting the running pipeline in the exploration stage. Thus, the cognitive app service has already read out the description of the corresponding stream enrichment pipeline from the pipeline repository and deployed the required nodes from node repository with their specific configuration parameters on the underlying big data infrastructure. Since there is no machine learning involved in this use case, the model repository is not active.

Overall, the stream enrichment pipeline consists of three core nodes (*Find neighbour Envisat pixels*, *Find intersections with ATKIS polygons*, *Pick Top K Land Use Classes*), that are using a set of seven queues (*wtstation*, *envisat*, *atkis*, *envisat-convolution*, *atkis-intersection*, *topk*, *topk-rgate*). The input data sources consists of (1) a data stream of sensor readings from different types of weather stations (vector data), (2) Envisat LST scans (raster data), as well as (3) ATKIS land use classes (vector data). At the beginning, the sensor readings need to be normalized based on a set of transformations according to the semantic description in the data source repository. Then, the normalized stream is consumed by the *first node* in combination with the latest Envisat LST scan of this region in order to find the neighbour Envisat pixels and calculate the convolution. The *second node* uses this output and computes the intersection with the land use class polygons of this region from ATKIS. Lastly, the *third node* selects the top-k land use classes and publishes them on a dedicated output queue. From there, the enriched output stream is stored in a database to persist the results and is presented to the user in a visual analytics view, thus allowing her to dictate top-k parameter changes to the running pipeline through the refinement gate input queue of the third node.

As shown in Table 1 the Stream Enrichment use case does not involve batch processing or learning and as consequence not all aspects of the use cases presented in Section 3 are discussed here in detail. However, with corresponding node design and/or pipe-line self-subscription loop such capabilities are harmonically supported by our framework.

7 CONCLUSION AND FUTURE WORK

In this paper, we have presented a conceptual architectural framework for spatio-temporal analytics at

scale. The work is motivated by the big data induced requirements in the field of geographic information systems. Due to the constant progress particularly in geoinformatics and the open source movement in big data, a sustainable approach is necessary to protect previous investments in technologies and operational effort and to prepare for future developments. The core challenges to sustainability that the architectural framework is facing are threefold: (1) heterogeneity of available data sources, (2) heterogeneity of use cases as well as, (3) heterogeneity of the big data landscape. These challenges are mainly addressed by an integrated and unified approach that builds on the established pipes and filters design pattern in combination with the continuous refinement model in BigGIS.

Our future work will on one hand focus on implementing the proposed conceptual architectural framework for the given uses cases and presenting these incarnations. On the other hand we try to extend the flexibility gained by the proposed architectural framework from the conceptual to infrastructure level by leveraging container technology for deployment and management of BigGIS.

ACKNOWLEDGEMENTS

The project BigGIS (reference number: 01IS14012) is funded by the Federal Ministry of Education and Research (BMBF) within the frame of the research programme “Management and Analysis of Big Data” in “ICT 2020 – Research for Innovations”.

REFERENCES

Apache Foundation (2016). Apache NiFi Documentation. https://nifi.apache.org/docs.html.

Berthold, M. R., Cebron, N., Dill, F., Gabriel, T. R., Kötter, T., Meinl, T., Ohl, P., Sieb, C., Thiel, K., and Wiswedel, B. (2007). KNIME: The Konstanz Information Miner. In *Studies in Classification, Data Analysis, and Knowledge Organization (GfKL 2007)*. Springer.

Buschmann, F., Henney, K., and Schmidt, D. C. (2007). *Pattern-Oriented Software Architecture – A Pattern Language for Distributed Computing*. John Wiley & Sons, New York.

Crossland, M. D., Wynne, B. E., and Perkins, W. C. (1995). Spatial Decision Support Systems: An Overview of Technology and a Test of Efficacy. *Decis. Support Syst.*, 14(3):219–235.

Eclipse Foundation (2016). GeoTrellis Documentation. http://geotrellis.io/documentation.html.

Emanuele, R. (2016). GeoTrellis landsat tutorial project. https://github.com/geotrellis/geotrellis-landsat-tutorial.

Esri (2016). ArcGIS and Big Data. http://www.esri.com/products/arcgis-capabilities/big-data/arcgis-and-big-data.

Guazzelli, A., Zeller, M., Lin, W., and Williams, G. (2009). PMML: An open standard for sharing models. *The R Journal*, 1(May):60–65.

Kreps, J. (2014). Questioning the Lambda Architecture. https://www.oreilly.com/ideas/questioning-the-lambda-architecture.

Marz, N. and Warren, J. (2013). *Big Data: Principles and Best Practices of Scalable Realtime Data Systems*. Manning Publications.

OGC (2013). Big Processing of Geospatial Data. http://www.opengeospatial.org/blog/1866.

Ord, J. K. and Getis, A. (1995). Local spatial autocorrelation statistics: Distributional issues and an application. *Geographical Analysis*, 27(4):286–306.

Peng, Y. and Liangcun, J. (2014). BigGIS: How big data can shape next-generation GIS. In *3rd Int. Conf. on Agro-Geoinformatics (Agro-Geoinformatics 2014)*, pages 1–6. IEEE.

Riemer, D., Kaulfersch, F., Hutmacher, R., and Stojanovic, L. (2015). Streampipes: Solving the challenge with semantic stream processing pipelines. In *Proc. of the 9th ACM Int. Conf. on Distributed Event-Based Systems*, DEBS ’15, pages 330–331, New York, NY, USA. ACM.

Sacha, D., Stoffel, A., Stoffel, F., Kwon, B. C., Ellis, G., and Keim, D. A. (2014). Knowledge Generation Model for Visual Analytics. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1604–1613.

Thakur, G. S., Bhaduri, B. L., Piburn, J. O., Sims, K. M., Stewart, R. N., and Urban, M. L. (2015). PlanetSense: A Real-time Streaming and Spatio-temporal Analytics Platform for Gathering Geo-spatial Intelligence from Open Source Data. In *Proc. of the 23rd SIGSPATIAL Int. Conf. on Advances in Geographic Information Systems*, pages 11:1–11:4. ACM.

Tomlin, C. (1990). *Geographic information systems and cartographic modeling*. Prentice Hall series in geographic information science. Prentice Hall.

Wiener, P., Stein, M., Seebacher, D., Bruns, J., Frank, M., Simko, V., Zander, S., and Nimis, J. (2016). BigGIS: A Continuous Refinement Approach to Master Heterogeneity and Uncertainty in Spatio-Temporal Big Data (Vision Paper). In *24th ACM SIGSPATIAL Int. Conf. on Advances in Geographic Information Systems (ACM SIGSPATIAL 2016)*.