Data Management in Multi-Agent Simulation Systems

From Challenges to First Solutions

Daniel Glake[1] Fabian Panse[1] Norbert Ritter[1] Thomas Clemen[2] Ulfia Lenfers[2]

Abstract: Multi-agent simulations are an upcoming trend to deal with the urgent need to predict complex situations as they arise in many real-life areas, such as disaster or traffic management. Such simulations require large amounts of heterogeneous data ranging from spatio-temporal to standard object properties. This and the increasing demand for large scale and real-time simulations pose many challenges for data management. In this paper, we present the architecture of a typical agent-based simulation system, describe several data management challenges that arise in such a data ecosystem, and discuss their current solutions within our multi-agent simulation system MARS.

Keywords: Multi-agent simulations, Spatio-temporal data, Polyglot data management

1 Introduction

In the digital age, more and more data is available used to predict future conditions and effects emerging from potential (re)actions. A popular approach to make such predictions are simulation systems. They can be used to predict the course of catastrophic events, such as social-ecological changes [LWC18], nuclear disasters [Wa18] or epidemics [ZKC05] (e.g., to play through the effects of various measures), but can also be used to control, predict and evaluate everyday aspects, such as traffic with climate influence, topographic changes and individual-driven decision-making [WGC18]. One way to simulate such complex social-world processes is to use a multi-agent simulation (*MAS*) [WR15] in which the system models every individual by a separate agent interacting directly or indirectly with other agents or the considered world. Since MASs are temporal systems and often deal with spatial properties given by the represented world and locations of simulated objects, spatio-temporal data management is a crucial part of modern MAS systems such as GAMA [Gr13], NetLogo [WR15] or MARS (Multi-Agent Research and Simulation) [We19]. Due to the ongoing digitalization (e.g., through the Internet of things) and the growing availability of data (e.g., open data), simulations receive more and more attention while the heterogeneity and volume of useful data are continually growing. For short-term planning, such as city-wide traffic-jam forecasting [We19], simulation results must be

[1] Universität Hamburg, Vogt-Kölln-Straße 30, 22527 Hamburg, Germany {glake,panse,ritter}@informatik.uni-hamburg.de
[2] HAW Hamburg, Berliner Tor 7, 20099 Hamburg, Germany {thomas.clemen, ulfia.lenfers}@haw-hamburg.de

doi:10.18420/btw2021-22

determined and aggregated quickly to provide value for decision support. In contrast, for long-term planning, e.g., infection spreading estimations [Ye06], the data management and simulation must be robust and offer sufficient capacity. Global sensitivity analyses further intensify these requirements. These circumstances cause several challenges in the data management of MAS systems, which we address in this paper from a general perspective before discussing some first solutions currently implemented in the MARS system.

This paper is structured as follows: In Section 2, we describe the typical components of a MAS system and how they are involved in the system's data management. Thereafter, we discuss open challenges for different data management aspects in Section 3 and describe in which way we address these challenges in MARS in Section 4. Finally, we discuss related work, conclude our paper and give an outlook on open research in Section 5.

2 Multi-Agent Simulation Systems

In this section, we describe the typical architecture of a MAS system (see Figure 1) and describe the individual components that interact within such a system. The architecture contains four main (represented by solid frames) and several optional (represented by dashed frames) components.

Simulation: The simulation component is the core of a simulation system. It receives a simulation model selected by the user and then loads all relevant input data from the data management component into the simulation's class model via the input adapter or the query mediator (see below). The simulation data can be categorized into four basic classes: Vector layers, graph layers, raster layers, and objects (agents and entities). Vector layers contain spatial information such as the position and structure of buildings, streets, or squares. Graph layers represent networks, for example, to model roads or public transport routes such as metro lines. Raster layers divide the considered space into equally large cells and store one or multiple – usually numerical – values per cell (e.g., the amount of rainfall). Agents are the active components of a simulation. Based on their environmental data, they are capable of autonomous actions and interact with each other to coordinate them. Such multi-agent interactions take place either directly via messages or indirectly via an environmental layer. Entities are not active, i.e., they cannot make decisions and initiate actions stand-alone, but have a life-cycle and can be used by agents (e.g., a car driven by a person).

After the initial state of the simulation has been created, the component starts the time-discrete simulation process. During this process, the simulation state is subject to a continuous change in each expiration of a previously defined step size (a so-called *tick*) with an optional real-time reference, e.g., layer values can change or agents can move. At the end of each tick, the current simulation state (including objects, layers, and tick metadata) is collected and passed to the output adapter, which forwards these results to their respective output channels (see below). State changes of individual agents, entities, and layers are synchronized to enable a consistent world state, i.e., all of them are always in the same tick. If the system

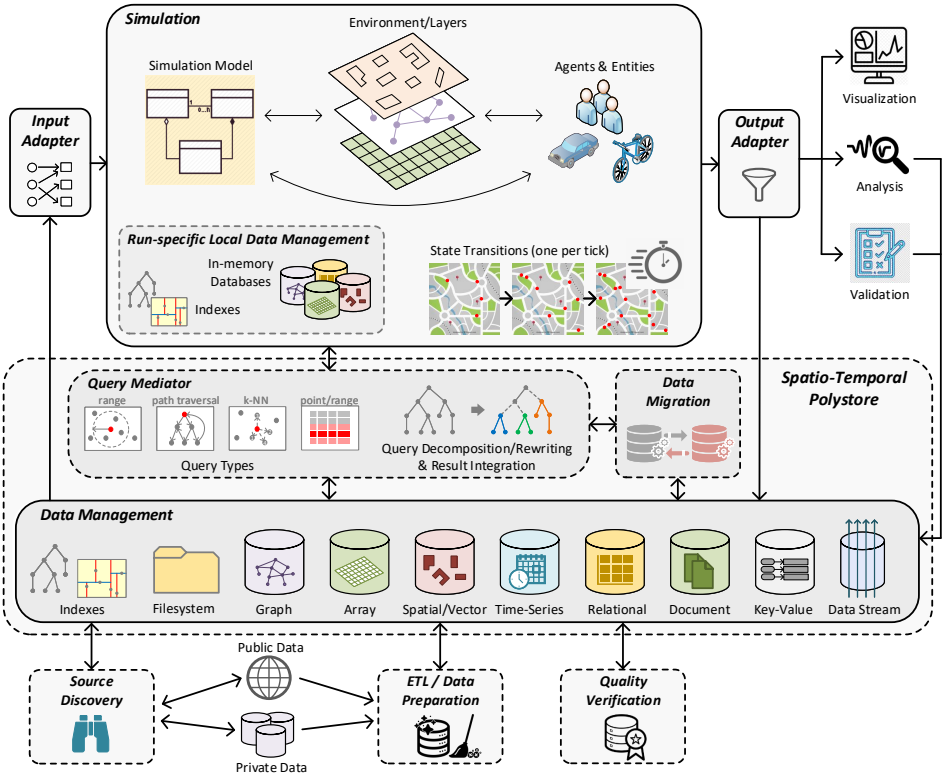

Fig. 1: Architecture of a Multi-Agent Simulation System

cannot load new data during runtime, the calculation of the following state can always only be based on the current one.

Data Management: The data management component includes various database systems of different data models, all of which serve a specific purpose. The user-defined agent-based models and some additional input files are persisted in a simple file system. Spatial rasters, e.g., for location-specific weather information, are suitable for data stores with array support (e.g., SciDB, Postgres, or Oracle GeoRaster). Vector-based features[3], including houses, factories, and other points of interest, are stored in databases supporting spatial indexing (e.g., PostGIS or MongoDB). The temporal data management (e.g., Timescale or InfluxDB) concerns the validity periods and transactions of data objects as well as aspatial time-related data, e.g., business hours. Single or combinations of aggregate-oriented NoSQL or relational systems with graph abstraction or mapping collection, as applied by multi-model databases (e.g., ArangoDB or OrientDB), are suitable for storing domain data. This domain includes

[3] A vector-based feature is a spatial geometry associated with a set of attributes and values.

entities and agents, each comprising subsets of value- or (un)directed reference-typed attributes (1:1, 1:n, or n:m) along inheritance hierarchies. Other facts or validity checks and analysis results can be stored in it as well. Finally, the input to the simulation system can be a data stream containing real-time data originating from sensor systems, such as temperature or air quality measurements. Additionally, the data management component manages several indexes, such as kd-trees for vector layers, that allows the input adapter and the query mediator (see below) fast access to the data.

The data management component has three subcomponents: The source discovery component is responsible for automatically detecting new and relevant datasets. Such a search can occur within a specific intranet (e.g., cloud), but also in the World Wide Web. The ETL/data preparation component loads external data (from private systems or the Web) into the data management component. Before loading it into an appropriate database, it standardizes, cleans, and enriches the data. The quality verification component is responsible for the high quality of the data already in the system. It includes a cross-database verification to detect inconsistencies between separately stored datasets.

Input Adapter: The input adapter supplies the runtime system with the simulation model (usually loaded from the file system) and initializes the first state as configured by the investigated scenario. This data mapping needs to overcome the impedance mismatch of the input data to the different kinds of models supported by the simulation component (i.e., vector, raster and graph layers as well as agent and entity classes). Therefore, it loads the data into local (in-memory) databases and indexes, kept for the duration of the currently running simulation process and – if implemented – are frequently updated by posing queries to the mediator (see below). These local databases allow fast access on layers, agents and entities, but limit the support only to point queries and k-NN queries with range filters. Unlike the query mediator, the input adapter is only used to map and load data to build the initial state of the simulation by utilizing user-defined scripts.

Query Mediator: In contrast to the input adapter, the query mediator enables dynamic and flexible ad-hoc access to the data management component by abstracting the required operations via a logical single query interface. Based on the mapping between defined operations and the underlying stores, the mediator decomposes queries into several subqueries while utilizing available store-specific features to make the most of their unique advantages. The subqueries are rewritten into native queries of the addressed stores and passed to them. Finally, their results are merged and forwarded to the running simulation or the user applying the analytical task. A rich query interface (including spatio-temporal operations and result formats), knowledge of store-specific features as well as planning, decomposing and rewriting queries are essential aspects in ensuring data transparency and providing short runtimes. The query mediator is supposed to support various types of queries, including: (i) spatio-temporal queries (e.g., to get objects from specific intersecting areas of Hamburg in the last month), (ii) range queries (e.g., to identify persons, cars, or buildings that are within a specific range to the ground zero of a disaster), (iii) k-NN queries (e.g., to identify the nearest bus stations or restaurants of a pedestrian), (iv) path traversals (e.g., to traverse

along the stations of a metro line), and (v) point queries (e.g., to access data of a particular point of interest). For practical reasons, k-NN queries must be combinable with range queries (i.e., the k-NN search is limited to a predefined range). Together with the data management and migration components, the query mediator forms a spatio-temporal polystore.

Data Migration: The structure and requirements for individual data objects change from time to time and often depend on the simulation processes using them. To meet such changing conditions, it can be useful to replicate data in different databases with different data models or migrate them from one model to another. Ideally, the system itself recognizes the demand for such a migration and automatically starts the corresponding migration process. Under certain circumstances, the query mediator initiates such a migration if it detects that the requested data are not available in the required format. In such a case, the migration must be performed at runtime either eagerly or lazily, for the current query only, or permanently. Since every migration step changes the location of the data, existing mappings between the databases and the simulation model may need to be updated.

Output Adapter: The output adapter is responsible to collect and forward snapshots of the individual objects and layers to the data management component and/or other software artifacts that aim to process them. Since the amount of data can be overwhelmingly large, exporting all of them can delay the simulation. Thus, it may be necessary that the adapter reduces the output to the most relevant values. It must also select a suitable format and compression method to keep the volume of data transferred as small as possible. Examples for relevant output data can be the volatile parts of the individual agents (e.g., position or vitality), but also the states of the different layers (e.g., temperature or water level) and additional measurements (e.g., traffic load).

Result Processing: Data exported by the output adapter can be stored directly in some of the databases, but can also be analyzed, visualized, and validated for violated constraints and expected behavior. The (often aggregated) results can, in turn, also be persisted in the databases. All three processing methods can be executed in real-time or batch mode, but only the first mode allows an intermediate intervention into the simulation's current state.

3 Challenges for Data Management

Besides the challenges that still need to be solved for polyglot data management in general [Pa16, Kr19, Ta17], such as query mediation [Cl98], automatic data migration [Kl16, SLD16] or cross-model replication [VSS18], there are a number of challenges that are specific to MAS systems. We will take a closer look at these challenges in this section.

Simulation Input: Although most simulations are limited to a specific area, it is almost always beneficial to transfer them to other areas by exchanging their location-specific data. For example when transferring a traffic simulation from Hamburg to Beijing, we need to exchange site maps, road networks and aspatial data, such as bus schedules. Since such a

location-based transfer is to be performed very often, flexibly and at short notice, adequate support in terms of (i) an *automatic discovery* and *acquisition* of relevant and qualitatively suitable input data, (ii) *preparation* of the newly acquired data, and (iii) an *automatic integration* of these data into the simulation model, is more relevant and crucial than in many other data integration use cases.

Dataset Discovery: Many spatio-temporal datasets are published in a structured form using a data portal/repository software, such as CKAN[4] [As20] or dataverse [Te20]. To find useful data in those portals, we must first discover a suitable data portal and then search for the required data in it. To support the second step, CKAN and dataverse provide several functionalities including full-text search and fuzzy matching on the datasets' meta data as well as browsing between related datasets. In contrast, finding a suitable data portal is – to the best of our knowledge – currently not supported by any software. If we do not find the required data in any portal, we have no choice but to crawl the World Wide Web and to extract the – often unstructured – data from the found websites [Fu13, Fe14, Fa18].

Preparation: After loading the data from the discovered sources, we restrict them to the spatio-temporal range of the simulation by removing all data points that are outside the area and time period of the corresponding scenario. Thereafter, we need to standardize, clean and enrich the remaining data. Spatial standardization includes transformations into the same spatial reference system, such as UTM or USNG. Raster layers have to be transformed to the same scale and converted so that their cells are congruent. Graph layers need to be transformed into compatible graph models [AG08]. Timestamps have to be normalized by transforming them into the same format, calendar and time zone. Finally, the attribute data can be standardized using conventional preparation techniques [Py99, KJN20]. The data cleaning has to include a removal (or repair) of (i) spatial [CS06, KL17] and temporal [Gu14, Zh17] outliers, (ii) inconsistencies between different polygons (e.g., overlapping borders) or timestamps (e.g., a building was demolished before it was built), and (iii) errors in the attribute data, such as typos or violated dependencies [GS13, IC19, Ch14] where some of these errors can only be detected by comparing data from different layers. Useful examples of data enrichment include using alternative data sources to refine the road network of Open Street Map (OSM) [RFS16] or applying geocoding to locate address data in the spatial layers accurately [CCW04].

Integration: After collecting and preparing relevant source data, we need to integrate them into the simulation model. This includes resolving conflicts in the sources' spatial and temporal overlaps, such as contradicting geographical details (e.g., the same building is represented by different polygons). The integration can be materialized or virtual [LN07, DHI12]. In the first case, we initially integrate all source layers of the same type (raster, vector or graph) into a single layer persisted in the simulation system. Thus, all data conflicts are resolved in a typical ETL process [KR02] once before the simulation process starts. This approach, however, is static and cannot deal with real-world changes that happen when

[4] Comprehensive Kerbal Archive Network

the simulation process is already running (see stream based input described below). In the second case, we integrate the source layers (and thus resolve conflicts) at query time by defining a global-as-view mapping [DHI12]. This is computationally more expensive, but flexible to changes against the source data. The same integration principles apply to temporal dimensions when we have several data sources covering different periods of time of the same spatial areas. To the best of our knowledge, there is currently no research that addresses a virtual integration of spatio-temporal data layers [GRC20].

Simulation Output: Exporting the snapshot of the current tick quickly becomes a bottleneck if the simulation is using a large number of agents. The biggest challenge is therefore to export the data without blocking the simulation process or creating a significant delay that would make it impossible to analyze, validate or visualize them in real time. The volume of the exported data is significantly related to whether we export only data changes or entire snapshots. The former reduces the volume, but makes immediate (possibly real-time) analyses, validations and visualizations of the simulation much more difficult because we need to reconstruct the actual snapshots from the exported changes.

Stream Based Input: The computed simulation states, including agent attributes and environment information, become fuzzier in their correctness as the simulation progresses and reaches further into the future. Public sensor data systems and APIs, such as the widely used SensorThingsAPI standard [LHK16], offer updates for temporally available environmental and entity-level information that can be used to reduce the corridor of uncertainty. Frequently updating the simulation states according to sensor-based input data by synchronizing the simulation with the real-time, results in a digital representation of real-world scenarios suitable for short-term forecasts and simplified global views of real-life happenings (e.g., to identify superspreading events within a pandemic). Problems are scalable handling of massive push-based inputs [WRG19] and merging incoming values on different time and granularity levels [CV86] without producing unrealistic simulation behavior (e.g., a full car park is emptied by beaming cars to remote locations). The latter can be done by either introducing them into the current simulation or forking a new one with the corrected state. Particularly relevant is the identification of model-independent growing uncertainties under consideration of user-defined constraints, defined via windowing queries and evaluated at the simulation's runtime. In addition to the usual integration problems, stream-based data present specific problems in dealing with non-equidistant inputs, duplicates, erroneous or noisy values, and sharp peaks. Solutions include the application of Kalman filters with wavelet corrections, comparisons of running windows for duplicate detection [SZ08, DNB13], and sliding aggregate functions [CV86]. For example, continuously averaging the attribute values of particular vector-based features can correct the simulation step by step.

Spatio-Temporal Query Interface: Because of the time-based definition of simulations, temporal operators are an essential aspect in the mediator's query interface. As it has been discussed by Siabato et al. [Si18], the support for Allen's interval operator [Al83] is essential for interval-based logical reasoning on time-series, getting versioned model objects. In context to the spatial characteristic, agents need access to environmental information for their

own decision making. This often corresponds to their current location,which requires k-NN queries with range filters. Such queries often have a circular shape, but can be abstracted to any geometrical shape. Polygon-based intersections require a pre-triangulation task to check for containment of the polygons' coordinates. Data access has to be provided by operators, such as *include*, *overlap* or *adjacent*, which also need to be part of the query interface [GRC20]. Since not only users perform analytical queries, but also the active agents themselves, the simulation can control queries against the mediator in time.

Spatio-Temporal Query Planning: The polystore has to manage the mapping between the simulation model's instance and its cross-system representation in the databases. This mapping requires a cross-system perspective, including requirements from the applied operations of the active agents in the simulation and subsequent analysis of results by the user. Populating the model with data from the polystore should be *independent* of the underlying databases and therefore transparent in the selection of convenient stores. Polyglot data storage offers the potential to meet a large set of (non-)functional requirements by taking into account the respective capabilities of each connected store in the mediated data processing. In order to exploit this potential for simulations, it is necessary to know the respective spatial, aspatial and temporal features of the individual databases and to describe them in a structured way. This description has to contain an input specification including constraints on expected objects as well as potentially produced outputs and their limitations. Query planning utilizes these feature descriptions to compute plans for distinct spatial and temporal queries, in which constraints are primarily affected by the expected models, for example, for relational data processing [SLD16]. Therefore, plans must address minimal intermediate migration steps where the cost of transfer does not exceed the cost of data processing. However, finding an optimal migration plan is NP-hard and can only be approximated [Kr19]. Beside migration problems, the system has to resolve references between data objects by providing an integration of partial results, either by implementing the bind-join [Ko16] approach or applying spatial-joins according to their references.

Further challenges concern cross-model data matching/merging [DHI12] and data lineage [HDB17] (e.g., to debug the simulation in case of errors).

4 Current Implementation

To meet the challenges of Section 3, we are extending our existing MARS architecture [Gl17, WGC18, We19], which aims to provide large scale, agent-based simulations for any domain expert. The key idea behind MARS is to combine the flexibility of self-adaptive and data model agnostic simulation systems, by following an as-a-service perspective on modeling and simulation (MSaaS). MARS schedules and runs simulation (or optional other agent-systems) pods within a heterogeneous cluster environment and scale-up and out along available processing units and computing nodes.

Simulation Input: For the integration of new input data, MARS provides an external Python subsystem called ODDI [5] [Gl19]. ODDI integrates a CKAN, Open Data Protocol, and Open Street Map client. The system utilizes a keyword search on the portals and retrieves all metadata by using the MinHash similarity. Results include vector, raster and table meta data, loaded in memory through OpenGIS Web services (WFS, WCS for vector and WMS for raster data)[6]. ODDI can also be used to audit the datas' quality through statistic analyses using a small statistics package and integrated plotter. New spatial data is prepared by transforming it into the WGS:84 EPSG:4326 reference system. Timestamps are uniformly converted to the same format. To integrate spatio-temporal data, MARS uses a hybrid approach. While spatial data layers are integrated materially, temporal changes are integrated virtually (i.e., we manage a separate layer for every time period).

Simulation Output: Since the focus of MARS are large-scale scenarios, the calculated results are proportional to the dimensioned agent types with their respective number of instances per simulated tick [WGC18]. The system persists snapshots of agents and layers along the underlying databases according to the current workload. Collected snapshots are persisted either as complete object versions or as deltas from the last versions. Each persistence task is applied in a specified output frequency or if a model object has been changed since the last tick. In the data management component, it can be decided whether the results are fully replicated in all data stores, a subset of data stores is used in order to produce specific output formats, or all data are saved only in one store or file format. Due to wrong or missing semantics in the resulting data, not all output combinations are possible for every layer or agent type (e.g., in the case of a missing support for matrix types or raster files). In addition, the output can be reduce to specific states by using predefined *output conditions* (e.g., the current spatial extension coming from the visualization on the client map). In order to enable a fast and parallel transfer of the output data to analysis, validation and visualization tools, the data are passed to a Kafka pipeline.

Modelling & Querying: MARS uses a polyglot approach to data modeling. The entire platform supports the complete workflow of simulative analyses and offers the external static-typed MARS DSL modeling and query language [Gl17]. The language includes a type inference system and links the agent-based paradigm with the spatio-temporal layer. The language conceptualize type definitions (agent, entity, vector- and raster-layer) and allows spatial queries by applying *conditional area* filters and *k-NN* queries as well as access on *time series* by specifying concrete points in time.

When comparing our current solution to the challenges described in Section 3, the following differences become apparent: (i) The system accesses the data management component only via the in- and output adapter. (ii) No requests are delegated to the database by a mediator. Data is kept entirely in-memory during the simulation. (iii) Spatio-temporal queries are limited to in-memory indexes. (iv) Query planning is considered at compile-time and does not include online data migration. (v) ODDI allows for automated retrieval of public data and

[5] Open Data Discovery and Integration
[6] WFS = Web Feature Service, WCS = Web Coverage Service, WMS = Web Map Service

spatial linking, but the datas' quality has to be checked manually. (vi) Streaming data into a running simulation is currently not supported. We plan to fill these gaps by extending MARS to a spatio-temporal polystore [Ta17]. In developing the mediator, we plan to use the MARS DSL as the logical query interface. We intend to implement data migration by adapting current approaches [Kr19, HKS19] to our needs, including an extension to spatio-temporal features. Query planning is firstly accomplished by pushing-down operations to store-related features, applying selection queries for spatial or temporal data and integrating results via bind-joins. We will leverage existing research on web data extraction [Fa18] and data cleaning [Ch14] to improve ODDI. To realize an integration of data streams into a running simulation, we plan to evaluate several strategies for adapting simulation states to these real values, without producing significant anomalies in the simulation behaviour.

5 Related Work & Conclusion

The main goal of MARS is to support large-scale scenarios for general purposes by utilizing polyglot data management with spatial and temporal data processing. Other existing simulation systems, such as NetLogo [WR15] and GAMA [Gr13], focus on smaller-scaled scenarios with less complexity or involved agents. Although GAMA offers direct SQL database access to its agents, it does not consider a polyglot design and keeps transparency on the level of the SQL language. Yang et al. [Ya18] also use the layer concept for simulations, but do not consider temporal changes of spatial objects. The system of Zehe et al. [Ze16] involves multi-store data management and attempts to use each store appropriately for the tasks at hand, but lacks in making these decisions transparent and generic by integrating an automatic query planning component. In addition, the system does not allow spatio-temporal queries. Existing multi-/polystore systems, such as RHEEM [Al19], Myria [Wa17], Polybase [De13] and ESTOCADA [Al19], follow a general-purpose approach by unifying the query-interface or applying intermediate (self-defined or automatic) migration steps between stores, providing uniform read-only access. In our opinion, this approach is unsuitable for simulations, because it ignores change operations, processing queries with store-specific features, which is a major challenge in polyglot data management [Pa16, Ta17], and capabilities for querying spatio-temporal data. Systems, such as CloudMdSql [Ko16] and BigDAWG [Du15], are first promising candidates. CloudMdSql provides users with direct access to native data storage languages by embedding them into a SQL-like language, but lacks in providing data independence, so that the user must always know which data is stored in which data store. BigDAWG provides transparency at the level of multiple query languages, but does not support any kind of updates.

In this paper we presented the general architecture for data management in spatio-temporal multi-agent simulations. We concluded a number of challenges and gave a brief overview of the current status with open issues of our own system MARS. Future work will address the development of a query mediator as well as the (further) development of components for data migration, quality management, and real-time processing.

References

[AG08] Angles, Renzo; Gutiérrez, Claudio: Survey of Graph Database Models. ACM Comput. Surv., 40(1):1:1–1:39, 2008.

[Al83] Allen, James F: Maintaining Knowledge about Temporal Intervals. Communications of the ACM, 26(11):832–843, 1983.

[Al19] Alotaibi, Rana; Bursztyn, Damian; Deutsch, Alin; Manolescu, Ioana; Zampetakis, Stamatis: Towards Scalable Hybrid Stores: Constraint-Based Rewriting to the Rescue. In: Proceedings of the International Conference on Management of Data (SIGMOD). Association for Computing Machinery, pp. 1660 – 1677, 2019.

[As20] Association, CKAN: , CKAN – The Open Source Data Portal Software. `https://ckan.org/`, 2020. [Online; accessed 12-12-2020].

[CCW04] Christen, Peter; Churches, Tim; Willmore, Alan: A Probabilistic Geocoding System based on a National Address File. In: Proceedings of the 3rd Australasian Data Mining Conference. 2004.

[Ch14] Chiang, Yao-Yi; Wu, Bo; Anand, Akshay; Akade, Ketan; Knoblock, Craig A.: A System for Efficient Cleaning and Transformation of Geospatial Data Attributes. In: Proceedings of the International Conference on Advances in Geographic Information Systems (SIGSPATIAL). ACM, pp. 577–580, 2014.

[Cl98] Cluet, Sophie; Delobel, Claude; Siméon, Jérundefinedme; Smaga, Katarzyna: Your Mediators Need Data Conversion! In: Proceedings of the International Conference on Management of Data (SIGMOD). ACM, p. 177–188, 1998.

[CS06] Chawla, Sanjay; Sun, Pei: SLOM: A New Measure for Local Spatial Outliers. Knowl. Inf. Syst., 9(4):412–429, 2006.

[CV86] Chair, Z; Varshney, PK: Optimal Data Fusion in Multiple Sensor Detection Systems. IEEE Transactions on Aerospace and Electronic Systems, (1):98–101, 1986.

[De13] DeWitt, David J.; Halverson, Alan; Nehme, Rimma; Shankar, Srinath; Aguilar-Saborit, Josep; Avanes, Artin; Flasza, Miro; Gramling, Jim: Split Query Processing in Polybase. In: SIGMOD. SIGMOD '13, Association for Computing Machinery, New York, New York, USA, p. 1255–1266, 2013.

[DHI12] Doan, AnHai; Halevy, Alon; Ives, Zachary G.: Principles of Data Integration. Morgan Kaufmann, 2012.

[DNB13] Dutta, Sourav; Narang, Ankur; Bera, Suman K.: Streaming Quotient Filter: A Near Optimal Approximate Duplicate Detection Approach for Data Streams. Proc. VLDB Endow., 6(8):589–600, 2013.

[Du15] Duggan, Jennie; Elmore, Aaron J; Stonebraker, Michael; Balazinska, Magda; Howe, Bill; Kepner, Jeremy; Madden, Sam; Maier, David; Mattson, Tim; Zdonik, Stan: The BigDAWG Polystore System. ACM SIGMOD Record, 44(2):11–16, 2015.

[Fa18] Fayzrakhmanov, Ruslan R.; Sallinger, Emanuel; Spencer, Ben; Furche, Tim; Gottlob, Georg: Browserless Web Data Extraction: Challenges and Opportunities. In: Proceedings of the International Conference on World Wide Web. ACM, pp. 1095–1104, 2018.

[Fe14] Ferrara, Emilio; Meo, Pasquale De; Fiumara, Giacomo; Baumgartner, Robert: Web Data Extraction, Applications and Techniques: A Survey. Knowl. Based Syst., 70:301–323, 2014.

[Fu13] Furche, Tim; Gottlob, Georg; Grasso, Giovanni; Schallhart, Christian; Sellers, Andrew Jon: OXPath: A Language for Scalable Data Extraction, Automation, and Crawling on the Deep Web. VLDB J., 22(1):47–72, 2013.

[Gl17] Glake, Daniel; Weyl, Julius; Dohmen, Carolin; Hüning, Christian; Clemen, Thomas: Modeling through Model Transformation with MARS 2.0. In: ADS@SpringSim. pp. 1–12, 2017.

[Gl19] Glake, Daniel; Weyl, Julius; Lenfers, Ulfia A.; Clemen, Thomas: SmartOpenHamburg Verkehrssimulation: Automatisierte OpenData Integration für Multi-Agenten Simulation mit MARS. In: Simulation in Umwelt- und Geowissenschaften. 2019.

[Gr13] Grignard, Arnaud; Taillandier, Patrick; Gaudou, Benoit; Vo, Duc An; Huynh, Nghi Quang; Drogoul, Alexis: GAMA 1.6: Advancing the Art of Complex Agent-Based Modeling and Simulation. In: PRIMA. pp. 117–131, 2013.

[GRC20] Glake, Daniel; Ritter, Norbert; Clemen, Thomas: Utilizing Spatio-Temporal Data In Multi-Agent Simulation. unpublished, 2020.

[GS13] Ganti, Venkatesh; Sarma, Anish Das: Data Cleaning: A Practical Perspective. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2013.

[Gu14] Gupta, Manish; Gao, Jing; Aggarwal, Charu C.; Han, Jiawei: Outlier Detection for Temporal Data: A Survey. IEEE Trans. Knowl. Data Eng., 26(9):2250–2267, 2014.

[HDB17] Herschel, Melanie; Diestelkämper, Ralf; Ben Lahmar, Houssem: A Survey on Provenance: What for? What form? What from? VLDB J., 26(6):881–906, 2017.

[HKS19] Holubová, Irena; Klettke, Meike; Störl, Uta: Evolution Management of Multi-model Data - (Position Paper). In: Heterogeneous Data Management, Polystores, and Analytics for Healthcare - VLDB Workshops, Poly and DMAH. Springer, pp. 139–153, 2019.

[IC19] Ilyas, Ihab F.; Chu, Xu: Data Cleaning. ACM, 2019.

[KJN20] Koumarelas, Ioannis K.; Jiang, Lan; Naumann, Felix: Data Preparation for Duplicate Detection. ACM J. Data Inf. Qual., 12(3):15:1–15:24, 2020.

[Kl16] Klettke, Meike; Störl, Uta; Shenavai, Manuel; Scherzinger, Stefanie: NoSQL Schema Evolution and Big Data Migration at Scale. In: IEEE Big Data. pp. 2764–2774, 2016.

[KL17] Kou, Yufeng; Lu, Chang-Tien: Outlier Detection, Spatial. In: Encyclopedia of GIS, pp. 1539–1546. Springer, 2017.

[Ko16] Kolev, Boyan; Bondiombouy, Carlyna; Valduriez, Patrick; Jimenez-Peris, Ricardo; Pau, Raquel; Pereira, José: The CloudMdsQL Multistore System. In: Proceedings of the International Conference on Management of Data (SIGMOD). ACM, p. 2113–2116, 2016.

[KR02] Kimball, Ralph; Ross, Margy: The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, 2nd Edition. Wiley, 2002.

[Kr19] Kruse, Sebastian; Kaoudi, Zoi; Quiané-Ruiz, Jorge-Arnulfo; Chawla, Sanjay; Naumann, Felix; Contreras-Rojas, Bertty: Optimizing Cross-Platform Data Movement. In: Proceedings of the International Conference on Data Engineering (ICDE). IEEE, pp. 1642–1645, 2019.

[LHK16] Liang, Steve; Huang, Chih-Yuan; Khalafbeigi, Tania: OGC SensorThings API Part 1: Sensing, Version 1.0. 2016.

[LN07] Leser, Ulf; Naumann, Felix: Informationsintegration - Architekturen und Methoden zur Integration verteilter und heterogener Datenquellen. dpunkt.verlag, 2007.

[LWC18] Lenfers, Ulfia A; Weyl, Julius; Clemen, Thomas: Firewood Collection in South Africa: Adaptive Behavior in Social-Ecological Models. Land, 7(3):97, 2018.

[Pa16] Papakonstantinou, Yannis: Polystore Query Rewriting: The Challenges of Variety. In: EDBT/ICDT Workshops. 2016.

[Py99] Pyle, Dorian: Data Preparation for Data Mining. Morgan Kaufmann, 1999.

[RFS16] Richter, Andreas; Friedl, Hartmut; Scholz, Michael: Beyond OSM – Alternative Data Sources and Approaches Enhancing Generation of Road Networks for Traffic and Driving Simulations. In: SUMO - Traffic, Mobility, and Logistics. Deutsche Zentrum für Luft- und Raumfahrt, pp. 23–31, 2016.

[Si18] Siabato, Willington; Claramunt, Christophe; Ilarri, Sergio; Manso-Callejo, Miguel Ángel: A Survey of Modelling Trends in Temporal GIS. ACM Computing Surveys, 51(2):1–41, 2018.

[SLD16] Schildgen, Johannes; Lottermann, Thomas; Deßloch, Stefan: Cross-System NoSQL Data Transformations with NotaQL. In: Proceedings of the 3rd ACM SIGMOD Workshop on Algorithms and Systems for MapReduce and Beyond (BeyondMR@SIGMOD). ACM, p. 5, 2016.

[SZ08] Shen, Hong; Zhang, Yu: Improved Approximate Detection of Duplicates for Data Streams over Sliding Windows. Journal of Computer Science and Technology, 23(6):973–987, 2008.

[Ta17] Tan, Ran; Chirkova, Rada; Gadepally, Vijay; Mattson, Timothy G: Enabling Query Processing across Heterogeneous Data Models: A Survey. In: IEEE Big Data. pp. 3211–3220, 2017.

[Te20] Team, Dataverse: , The Dataverse Project – Open Source Research Data Repository Software. `https://dataverse.org/`, 2020. [Online; accessed 12-12-2020].

[VSS18] Vogt, Marco; Stiemer, Alexander; Schuldt, Heiko: Polypheny-DB: Towards a Distributed and Self-Adaptive Polystore. In: 2018 IEEE International Conference on Big Data (Big Data). IEEE, New York, New York, USA, pp. 3364–3373, 2018.

[Wa17] Wang, Jingjing; Baker, Tobin; Balazinska, Magdalena; Halperin, Daniel; Haynes, Brandon; Howe, Bill; Hutchison, Dylan; Jain, Shrainik; Maas, Ryan; Mehta, Parmita; Moritz, Dominik; Myers, Brandon; Ortiz, Jennifer; Suciu, Dan; Whitaker, Andrew; Xu, Shengliang: The Myria Big Data Management and Analytics System and Cloud Services. In: Proceedings of the Conference on Innovative Data Systems Research (CIDR). 2017.

[Wa18] Waldrop, Mitchell: Free Agents - Monumentally Complex Models are Gaming out Disaster Scenarios with Millions of Simulated People. Science, 360(6385):144–147, 2018.

[We19] Weyl, Julius; Lenfers, Ulfia A; Clemen, Thomas; Glake, Daniel; Panse, Fabian; Ritter, Norbert: Large-Scale Traffic Simulation for Smart City Planning with MARS. In: SummerSim. pp. 1–12, 2019.

[WGC18] Weyl, Julius; Glake, Daniel; Clemen, Thomas: Agent-Based Traffic Simulation at City Scale with MARS. In: ADS@SpringSim. pp. 1–9, 2018.

[WR15] Wilensky, Uri; Rand, William: An Introduction to Agent-Based Modeling: Modeling Natural, Social, and Engineered Complex Systems with NetLogo. MIT Press, 2015.

[WRG19] Wingerath, Wolfram; Ritter, Norbert; Gessert, Felix: Real-Time & Stream Data Management - Push-Based Data in Research & Practice. Springer Briefs in Computer Science. Springer, 2019.

[Ya18] Yang, Liang Emlyn; Hoffmann, Peter; Scheffran, Jürgen; Rühe, Sven; Fischereit, Jana; Gasser, Ingenuin: An Agent-Based Modeling Framework for Simulating Human Exposure to Environmental Stresses in Urban Areas. Urban Science, 2(2):36, 2018.

[Ye06] Yergens, Dean; Hiner, Julie; Denzinger, Jörg; Noseworthy, Tom: Multiagent Simulation System for Rapidly Developing Infectious Disease Models in Developing Countries. In: MAS*BIOMED. pp. 104–116, 2006.

[Ze16] Zehe, Daniel; Viswanathan, Vaisagh; Cai, Wentong; Knoll, Alois: Online Data Extraction for Large-Scale Agent-Based Simulations. In: Proceedings of the 2016 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation. pp. 69–78, 2016.

[Zh17] Zhang, Aoqian; Song, Shaoxu; Wang, Jianmin; Yu, Philip S.: Time Series Data Cleaning: From Anomaly Detection to Anomaly Repairing. Proc. VLDB Endow., 10(10):1046–1057, 2017.

[ZKC05] Zaiyi, GUO; Kwang, HAN Hann; Cing, TAY Joc: Sufficiency Verification of HIV-1 Pathogenesis Based on Multi-Agent Simulation. In: GECCO. pp. 305–312, 2005.