Multi-Agent Systems and Digital Twins for Smarter Cities

Thomas Clemen
Nima Ahmady-Moghaddam
Ulfia A. Lenfers
Florian Ocker
Daniel Osterholz
Jonathan Ströbele
thomas.clemen@haw-hamburg.de
Department of Computer Science, Hamburg University of Applied Sciences
Hamburg, Germany

Daniel Glake
Department of Informatics, Universität Hamburg
Hamburg, Germany
daniel.glake@uni-hamburg.de

ABSTRACT

An intelligent combination of the Internet of Things (IoT) and approaches to modeling and simulation is one of the most challenging endeavors for future cities, manufacturing industries, and predictive maintenance. Digital Twins take on a unique role here. However, the question of what a Digital Twin is and what differentiates it from a regular model is still open. We present an experimental setup for integrating an existing simulation model of Hamburg's traffic system with the city's real-time sensor network. The Digital Twin is implemented using the large-scale multi-agent framework MARS. The entire process from the model description to retrieving real-time data from the IoT sensors and incorporating it in the simulation is presented. As a first prototypical example, a multi-modal mobility model was connected to real-world bike-sharing locations in Hamburg. We find that the combination of multi-agent systems and IoT sensors as a Digital Twin shows enormous potential for city planners, policy stakeholders, and other decision-makers. By correcting the course of a simulation via real-time data, the corridor-of-uncertainty that is intrinsic to some simulation models' use can be reduced significantly. Furthermore, any divergence of simulated and sampled data can lead to a deeper understanding of complex adaptive systems like big cities.

CCS CONCEPTS

• **Computing methodologies → Multi-agent systems**; **Simulation types and techniques**.

KEYWORDS

Multi-Agent Systems, Digital Twins, Internet of Things

ACM Reference Format:
Thomas Clemen, Nima Ahmady-Moghaddam, Ulfia A. Lenfers, Florian Ocker, Daniel Osterholz, Jonathan Ströbele, and Daniel Glake. 2021. Multi-Agent Systems and Digital Twins for Smarter Cities. In *Proceedings of the 2021 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation (SIGSIM-PADS '21), May 31-June 2, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3437959.3459254

1 INTRODUCTION

The increasing global trend toward living in cities emphasizes the imperative for intelligent tools and services from computer science, especially to facilitate urban planning, manage limited resources, and provide decision-support. In this context, Santana et al. [32] listed some critical technologies such as the Internet of Things (IoT), Big Data management, and Cloud Computing.

Later, Batty [2] added Artificial Intelligence (AI) and related topics as well as the concept of the Digital Twin to this list. Moreover, Wildfire [37] stated: "*There is no doubt that opportunities for massive-scale Digital Twins are significant, given their potential to bring great benefits to city and infrastructure clients such as improved asset maintenance, business transparency, and better and more extensive optioneering and decision-making.*"

Such statements remember the promising capabilities and applications of other modeling and simulation paradigms and technologies – for example, Agent-based Models (ABM) – in the context of Smart Cities [8]. Any initially high expectations of novel technologies that do not produce results fast enough might lead to a loss of potential when exploration and research of those technologies are retired prematurely. A precise definition and understanding of concepts and ideas is essential. Therefore, it is worthwhile investigating the differences between models and Digital Twins in general.

Rasheed et al. [31] proposed the following definition: "*A Digital Twin is defined as a virtual representation of a physical asset enabled through data and simulators for real-time prediction, monitoring, control, and optimization of the asset for improved decision making throughout the life cycle of the asset and beyond.*" A more formal definition is still lacking.

This paper suggests that Multi-Agent Systems (MAS) are especially well-suited for implementing Digital Twins. Additionally, we intend to analyze the related impact of streaming real-time data into simulation runs. We linked an existing large-scale ABM of the

City of Hamburg, Germany, to the IoT sensor network provided by the city's administration, creating a Digital Twin of a subset of the traffic system through which participants can navigate based on individual decisions and via multiple modalities. We introduce a conceptual and technical framework that grants software agents access to these sensors. A special focus lays on integrating spatio-temporal data objects into the state space of a simulation run. We explore the logical impact of this approach on the simulation outcome in a practical example and evaluate the potential value of the integration architecture for Smart Cities.

The paper is structured as follows: Section 2 provides a brief overview of the related work in this field. Section 3 offers the authors' perspective on Digital Twins and their conceptualization through ABMs, proposing a formal definition to distinguish between the two concepts. Section 4 describes the Digital Twin of Hamburg and its underlying dynamics as well as model architecture. Section 5 gives a conceptual and technical presentation of the integration architecture – the main contribution of this paper – and Section 6 elaborates on some implementation details. Section 7 introduces a proof of concept with some preliminary results. Section 8 discusses the results of the approach and a conclusion finalizes the paper with some thoughts for future work.

2 RELATED WORK

A good number of specialized, large-scale modeling and simulation frameworks for city traffic scenarios exists. MATSim [15] and SUMO [24] are prominent examples, each with a usage record spanning many cities globally. The underlying model structure is well-adapted to intermodal traffic solutions and allows for quick, special-purpose model development. However, the authors are not aware of existing software frameworks that cover the linkage between such a model and IoT sensors in a generalized way.

The idea of linking software agents and IoT technology is relatively recent. Neyama et al. [25] combine driving behavior in a vehicular IoT system with a MAS. Krivic et al. [21] propose the use of an agent-based approach for the discovery and management of IoT services. Zheng et al. [39] describe a universal smart home control platform architecture based on a MAS. Pico-Valencia and Holgado-Terriza [29] and Savaglio et al. [33] provide a good overview of systems and technologies in that context.

Another promising approach to describe the interaction between a simulation model and its physical counterpart is 'symbiotic simulation' [1]. Such simulation systems are designed to support decision-making at operational levels by making use of (near) real-time data generated by the physical system and streamed to the development of the simulation model [28]. This topic is linked to the definition of Digital Twins described later in this study.

Still, the ongoing utilization of real-world spatio-temporal data within simulation models holds many challenges. Clay et al. [9] and Kieu et al. [18] state that it is currently not possible to use ABMs for real-time simulation due to the absence of established mechanisms for dynamically incorporating real-time data.

3 DIGITAL TWINS

The term 'Digital Twin' (DT) was coined by Michael Grieves in the early 2000s [2]. Since its inception, the concept has been applied to various fields. It seems necessary to us to clearly separate the general term 'model' from the term 'Digital Twin'. We suggest to view a DT as a model with a close – and sometimes permanent – connection to the real system. Thereby, the model and the physical system might eventually not be readily distinguishable from each other. Batty [3] states that "*if our methods and models are now part of the system, they become part of the real thing*." Thus, not every model is a DT, but each DT comprises at least one model.

3.1 Agent-based Models as Digital Twins

Without loss of generality, an ABM is defined as a triple $ABM = \{A, E, L\}$ with A a set of agents, E a set of entities, and L a set of environmental layers. Figuratively, an ABM can be compared to a board game, where the set of layers comprises the board and the agents' software instances represent the tokens that act and react based on their intrinsic rules and algorithms. In such a game setup, an entity could represent an obstacle or a barrier.

Extending this definition, a DT can be defined as a quintuple $DT = \{A, E, L, S, M\}$ with S a set of real-world hardware sensors and M a mapping between $\{A \cup E \cup L\}$ and S. Here, the term sensor serves as a placeholder for any physical object that provides a software-accessible interface to the outside world. Depending on the nature of a given sensor, a mapping can consist of any combination of the three software concepts. Consider, for example, a sensor $s_{CO} \in S$ in the city of Hamburg that measures the level of carbon monoxide (CO) in the air. There is an entity object $e_{CO} \in E$ that represents a physical air quality sensors virtually in the DT. Thus, there exists a mapping $m_{CO} \in M$ with $m_{CO} : e_{CO} \rightarrow s_{CO}$. The entity e_{CO} is called the Digital Twin Instance (DTI) of s_{CO} with $e_{CO} := DTI(s_{CO})$. The mapping allows an agent $a \in A$ to determine the concentration of CO at the location of sensor s_{CO} by querying the entity e_{CO}.

Implementing a DT and its DTIs places some challenging requirements on the software frameworks used:

- *Autonomy.* IoT sensors sample their data independently. This autonomy is also demanded from its virtual counterpart.
- *Scalability.* In the near future, DTs of Smart Cities might incorporate very high numbers of IoT sensors and other physical objects that need to be represented virtually.
- *Robustness.* Sensors can fail or provide invalid values. A DT's logic needs to mitigate that in an appropriate manner.

3.2 Agents and Entities as DTIs

Generally, DTIs can be implemented using either the agent or the entity concept. In the context of Smart Cities, agents and entities representing physical assets can potentially be considered as their respective DTIs. When designing a DTI, an important consideration is its intended level of behavioral and functional complexity:

(1) The Passive DTI
This is the most basic case. The software agent or entity is connected to a physical sensor and incorporates state changes. Example: a human agent opens an umbrella when it starts to rain at its location.
(2) The Active DTI
In this case, the software agent is capable of actively changing the state of a physical asset. For example, a DTI of a

drone calls an emergency service when it detects a person lying on the ground.

(3) The Interactive DTI
This most sophisticated case is related to 'agent's embodiment', i.e., the physical asset and its DTI are so closely connected that they work together as a unit. Example: a DTI can 'ask' its assistance robot to touch an obstacle to learn about its surroundings, i.e., the agent utilizes the robot to interact with the environment through a physical body.

4 DIGITAL TWIN OF HAMBURG

As part of the SmartOpenHamburg project (SOHH), we developed a base model for urban mobility [35] in which agents are virtual representations of people that travel within the city using a variety of modalities – bicycles and cars, among others – that are modeled as entities. Layers provide an agent with the capabilities of exploring other agents, entities, and additional information that is relevant for the scenario at hand. We distinguish between the model description and a simulation scenario. The former provides the general structure and functionality of the model. The latter contains additional configurations and data sources for all included components of the DT. As only passive DTIs are described by this study, integrated data sources are read but never written, making for a unidirectional flow of data from the real world into the model.

4.1 Model Description

In order to integrate real-time data in the system, we created a model that uses only a subset of the available base model components and thus only a subset of modality choices, but is extended by a mapping between sensors and their respective model entities. Using the formal definition of a DT proposed in Sec. 3.1, the model is described as follows:

- A = { *CycleTraveler* }
- E = { *RentalBicycle*, *BicycleRentalStation* }
- L = { *SchedulerLayer*, *GatewayLayer*, *CycleTravelerLayer*, *BicycleRentalLayer* }
- S = { *RentalStationThing* }
- M = { m_{RS} : *BicycleRentalStation* $\longrightarrow$ *RentalStationThing* }

The model's protagonist is an agent named *CycleTraveler* which moves from a source to a destination coordinate on foot or, if feasible, by using a *RentalBicycle*. Agents are spawned by the *SchedulerLayer* that holds their temporal descriptions with, inter alia, their area of spawning and their area of destination. Within these areas, each agent's start and goal coordinate is chosen randomly.

An agent's movements occur on multiple spatial graph environments, each designated for a distinct modality (here: walking and cycling). The *SpatialGraphEnvironment* (SGE) [35] manages this graph, supervises all movement with respect to consistency (e.g. collision prevention), and allows exploration queries to collect information about surrounding agents. The SGE provides route searching capabilities and k-nearest neighbors (k-NN) queries to resolve nearest nodes. The spawning and destination areas may, however, exceed the SGE's extent. To reach the graph (or their goal outside of the graph), agents use entry (or exit) points, respectively. These points are defined in the *GatewayLayer* which contains gateway points between different sections of the city's travel network (e.g. suburban railway stations). A threshold is defined within which agents may enter the graph via the nearest node of the SGE. If an agent's distance to such a node is greater than the threshold, it enters the graph using the gateway point that is located closest to its goal. Similarly, an agent whose goal is beyond the graph's periphery uses the closest gateway point if the threshold is exceeded.

4.2 Model Dynamics

Within the simulation area, agents always start moving on foot. Each agent checks if switching to available transportation devices would make for a faster route to its destination. To evaluate routes and modality choices, an agent performs a static cost estimation for available alternatives and chooses the cheapest option with respect to travel time. In detail: available *RentalBicycle*s are located at *BicycleRentalStation*s that can be explored by querying the *BicycleRentalLayer*. The agent chooses to switch if at least one bicycle is available and if the time consumption for the detour to the RS and from the target RS to the goal is smaller than the expected time saved due to higher traveling speed. Bicycle rental occurs on a first-come-first-served basis; hence, renting a bicycle is possible only if a bicycle entity is available at the time the agent reaches the RS. The agent is fully optimistic and does not distinguish between a *BicycleRentalStation* with only a few available *RentalBicycle*s and one with multiple available *RentalBicycle*s. Therefore, it is possible that all previously available *RentalBicycle*s at a *BicycleRentalStation* are in use by the time an agent reaches it. The agent then has to continue on foot, either directly to its goal or to the next *BicycleRentalStation* with available *RentalBicycle*s. A new cost estimation will be performed, because the static estimation is only based on the current world state. Changing assumptions force the agent to replan.

In general, an agent is able to move with different transportation devices (hereafter vehicles) if it meets the requirements of that vehicle. We, therefore, introduced a so-called 'movement handle' concept to the model. Every vehicle specifies steering capabilities as prerequisites for using it. Upon trying to use the vehicle, an agent's capabilities are checked. If all requirements are met, a steering handle is provided and can be used by the agent to move the vehicle. Beside capabilities, a vehicle has further checks that might prevent an agent from using it, such as already being in use or its proximity to the agent.

Every vehicle entity has a set of properties that influences its movement dynamics. Vehicle types share the same algorithm for acceleration; bicycles are implemented using the Wiedemann algorithm [36]. These equations are embedded in a control system that accounts for traffic regulations. Analogically to steering, an agent requires passenger capabilities for co-driving a vehicle as well as a passenger handle to leave the vehicle at will. Walking is the fallback modality and is always available to an agent.

4.3 Model Integration

The DT connects to the physical world by an IoT concept, a *thing*. Figure 1 displays the ability of a thing rental station (RS) in the physical world to gather and forward information from its sensors to the entity *BicycleRentalStation* – its DTI in the model. Before being integrated in the model, however, the information undergoes a mapping phase. During this phase, the information is validated and sorted in a chronological catalogue. It is withheld from the DTI until the phenomenon time of the information (the timestamp from which the information is valid) and the simulation time intersect. The DTI may then integrate the information in its state. However, we implemented synchronization time points that notify a DTI to integrate the most up-to-date information into its state. Therefore, all DTIs are synchronized at the same time point and not individually, even if new information for a particular DTI might already be available.

There are different synchronization strategies that can be pursued in an effort to mitigate the corridor of uncertainty that tends to widen over the course of a simulation. The approach presented here aims for synchronization via integration of external real-time data in order to adapt to the ground truth. In this sense, data integration mechanism can broadly be categorized as soft and hard. In our model, a mixture of hard and soft synchronization is chosen by, on the one hand, directly updating the amount of bicycles at the RS (hard) and, on the other hand, leaving bicycles en route untouched (soft).

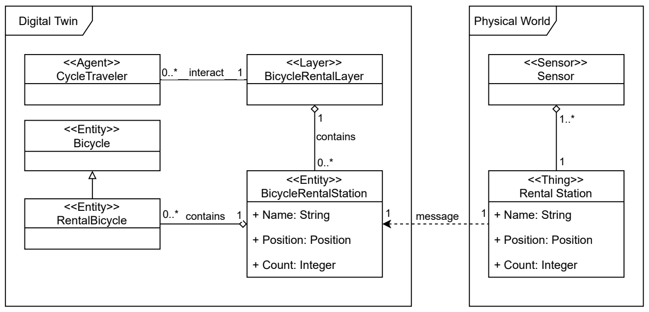

Figure 1: Design model showing the integration of bicycle rental stations and agents' interaction with them.

5 INTEGRATION ARCHITECTURE

Changes to the model can be triggered from within the model or from outside by the sensor infrastructure. These changes are categorized as either external changes over time (*spatial changes* applied within the spatial object) or internal changes over time (*temporal changes* triggered by a cascade of actions originating from agents or entities such as an agent's movement or interactions with spatial objects). Both types of change can alter the geometric instance evolutionary in shape, position, rotation, or statically in a respective data value [11]. Furthermore, they both conclude in the selection of subsequent actions by agents, propagating events independently so that new actions emerge and interact with spatial objects. For our model, these changes affect the decision and movement behaviour of *CycleTraveler* agents, adaptively guiding them to currently available bicycle resources.

5.1 Input Classification

Supporting a variety of input data and formats for divergent simulation purposes, our approach represents spatio-temporal resources by utilizing a *vector-layer* data structure. Such a layer provides geospatial querying and integration of temporal-dependent, external consulting dynamic changes. Therefore, this layer type manages a *multi-dimensional index* and provides constrained k-NN- and geometric queries [12] over shapes as well as a component, called *temporal catalogue*, to serve data changes over time. In addition to supporting widely used standard exchange formats, a *vector-layer* utilizes a multi-format adapter to consume and prepare different inputs. Figure 2 shows the classification of *vector-layer* inputs.

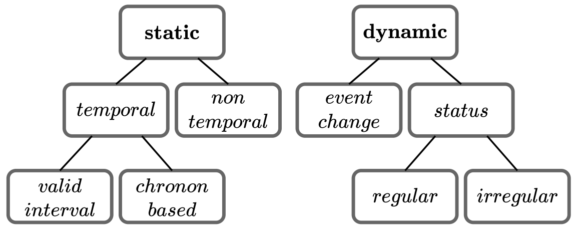

Figure 2: Input data classification for the temporal vector-layer.

Static inputs refer to, among others, all inputs such as files and queries that entirely exist at the start of a simulation or can be obtained during the initialization phase. *Dynamic* inputs include all data that are retrieved by push-based data stream systems and produced at runtime. Both input types are implicitly or explicitly temporal-dependent, based on their validity period. Further, the *vector-layer* distinguishes the objects as follows:

- Static non-temporal data have no time affinity and are valid for the whole simulation time.
- Static data objects may have a *valid interval* that specifies precisely the beginning and the exclusive end when the contained data values are considered as valid.
- Static data with an assigned single *chronon* time mark are exactly valid when the time point is valid.
- Messages from event-based streams for considered DTI depend on sporadically occurring real events and have no validity period. They are valid from the exact event time, called *phenomenon time*, and are invalid when the next event occurs.
- State-based streams are discontinuous input messages that return the last complete state of an observed value when the value has been changed. They are valid *regularly* by recurring fixed time intervals or *irregularly* by indefinite time intervals.

Beside discontinuous streams, there exist continuous streams in the real world. Since the approach focuses on discrete data integration, these analogue sources are not considered but can be extended further by quantizing the value and discretizing them in time.

5.2 Temporal Integration

All *static* and *dynamic* sources are equally relevant. This is because multiple data sets are heterogeneous, provide only a subset of required information for the considered scenario, and do not contain validity periods – specifically, only a temporal marker assigned by, e.g., corresponding sensors or occupied by the hub system time.

Utilizing a metadata description, the *vector-layer* refers to multiple *static* sources of data sets. Metadata entries contain implicit or explicit temporal references (such as a validity period or validity marker) applied to all data objects contained in the source. In contrast to *static* sources, *dynamic* sources are configured by the given scenario description via push-based queries. The system registers such a query at the real-time system and returns the complete state of an observed object when any matching element changes.

The system forwards data objects into two intermediate, distinct catalogue stores in order to encapsulate the potential heterogeneous sources from the model. The *non-temporal* catalogue contains values that are valid for the entire simulation time. The *temporal* catalogue preserves multiple groups with time series, each concerning a sequence of data changes. New time-series entries lead to the creation of new entities. An insertion of data to an existing sequence causes the invalidation of the previous oldest value to the valid time of the new one. This constructs valid periods of data inherently [20, 30].

Since external data are integrated, the approach needs to deal with *divergent transmission latency*, *unequal clocks* of internal and external scopes, and *duplicates*. This includes data cleaning by removing spatial [6, 19] and temporal [13, 38] outliers and errors in the attribute data, such as typos or dependency violations [7, 10, 17].

These categorical preparation tasks of input data take place on the syntactic and semantic level. According to public standards, the syntactic level ensures readability to all supported formats (CSV, GeoJSON, ESRI format, Raster-ASC, GraphML, GML2.0/3.0 and KML2.2) implemented by their respective providers. Any incorrect input is intercepted directly. At the level of semantics, the system checks spatial and temporal outliers of data objects by *intersection* with the simulation area and the considered simulation period. However, since some outliers serve a semantic purpose, this can be adjusted by the user based on the scenario.

Furthermore, the system builds a domain-independent solution by utilizing a spatial join over *intersection* of equal geometries, linking multiple spatial inputs for the same model object (e.g., a *BicycleRentalStation*). Associated domain data are unified, and conflicts can be resolved unilaterally along the validity period or must be prioritized by the user through the mapping. Therefore, the temporal catalogue acts as a cache for static or dynamic external inputs. This means that the system is transparent to the input. We can apply all data preparation tasks only by first mapping each external data object into an internal representation.

Figure 3 shows an example of input transformation, mapping *static* and *dynamic* data to two distinct stores named *temporal catalogue* (temporal-dependent objects) and *non-temporal catalogue* (objects without temporal reference). For incoming *static* data, the *vector-layer* builds a group with an abstract identifier (such as a) by assigning input data values (such as 2 or 6) in the form of ordered relations (such as $\{(a, 2), (a, 6)\}$) and according to their timestamp (such as t_a). For *dynamic* data, the value is associated by the subscribed topic (endpoint) URL (such as *url*2) and collected from consulting *phenomenon time* or by the *status interval* of considered things.

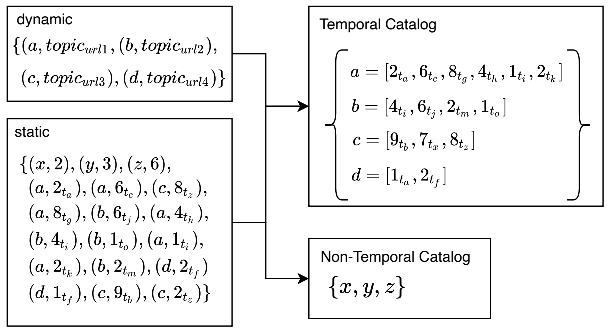
dynamic
$\{(a, topic_{url1}), (b, topic_{url2}), (c, topic_{url3}), (d, topic_{url4})\}$
static
$\{(x,2),(y,3),(z,6), (a,2_{t_a}),(a,6_{t_c}),(c,8_{t_z}), (a,8_{t_g}),(b,6_{t_j}),(a,4_{t_h}), (b,4_{t_i}),(b,1_{t_o}),(a,1_{t_l}), (a,2_{t_k}),(b,2_{t_m}),(d,2_{t_f}) (d,1_{t_f}),(c,9_{t_b}),(c,2_{t_z})\}$
Temporal Catalog
$a = [2_{t_a}, 6_{t_c}, 8_{t_g}, 4_{t_h}, 1_{t_l}, 2_{t_k}]$
$b = [4_{t_i}, 6_{t_j}, 2_{t_m}, 1_{t_o}]$
$c = [9_{t_b}, 7_{t_x}, 8_{t_z}]$
$d = [1_{t_a}, 2_{t_f}]$
Non-Temporal Catalog
$\{x, y, z\}$

Figure 3: Temporal transformation of input data to model object groups and static data sets.

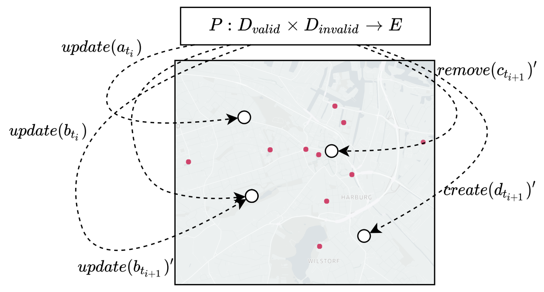

Figure 4: Vector-layer update and remove process.

The *vector-layer* implements a population function, forwarding updates or removing invalid model objects from the simulation scope at the end of their lifetime and checking for geometry changes. Figure 4 shows an exemplary population function P in which all valid (D_{valid}) and invalid ($D_{invalid}$) data objects are resolved to their corresponding entity $e \in E$ (as defined in Sec. 3.1) for the current and subsequent simulation time step t_i and t_{i+1}, respectively. For each object, we decide whether the entity is *created*, *updated*, or *removed*. The population of data considers the following cases:

- The value of a model object is now invalid and will be *removed* and, additionally, no other value is available. Therefore, the spatial index needs to be updated.
- The model object gets a flip *update*, preserving the geometry where the current assigned value is now invalid. But the object received a new value immediately, flipping from one value to another. The index needs no update.
- The new incoming value has changed the location, position, rotation (for non-point geometries), or scale. The *vector-layer* updates the index.
- The value is valid only for exactly one time point. Before the beginning of the next simulation time step, the *vector-layer* *removes* the feature and updates the index.

5.3 Technical Integration

Updates arrive from data sources outside of the simulation and must be integrated into the model accordingly. This integration includes the transformation of the external data schema into the internal model structure. The system derives the external data schema from

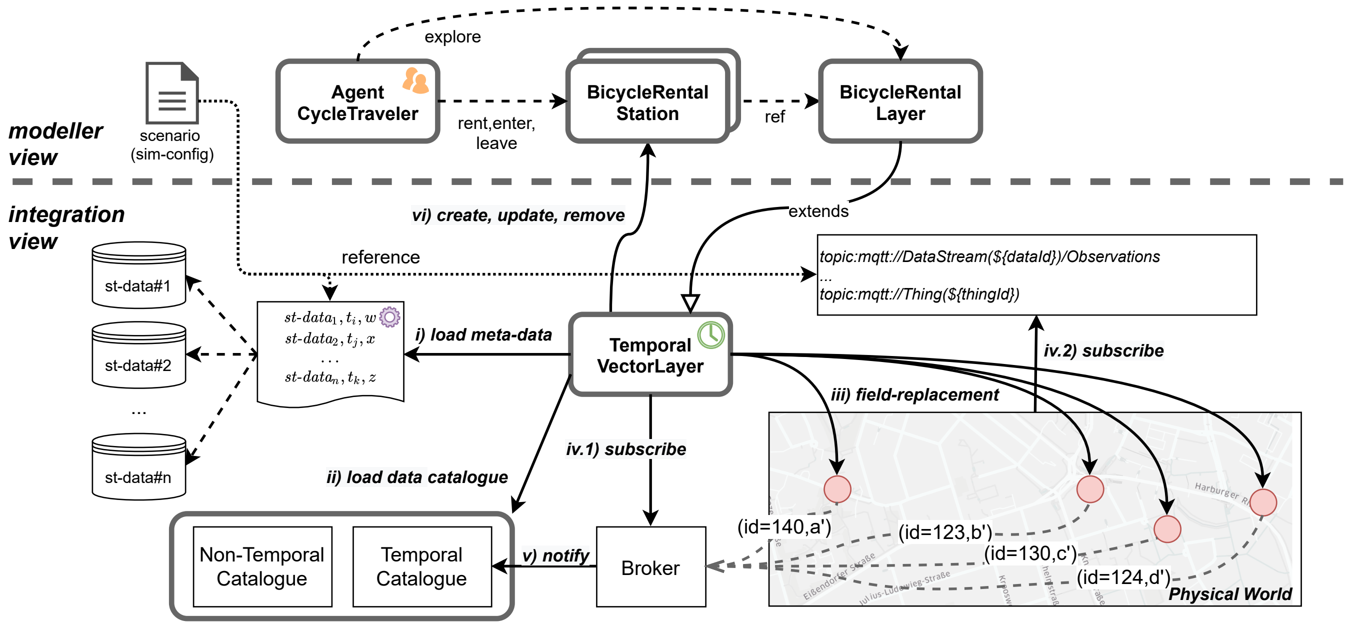

Figure 5: Technical view on the spatio-temporal integration of multiple real-time information sources.

the outgoing scenario and its configuration and implements several database connectors and mappers for this purpose, concluding in a polystore setting [11]. Each mapper allows queries to the underlying system and follows the corresponding interaction protocol. Currently, real-time data integration is supported by utilizing the SensorThings API [23] with its pub/sub model, using the widely used transmission protocol MQTT[1]. Furthermore, with COAP[2] [5], other push-based services are available. MQTT can also be used independently of the SensorThings API.

Figure 5 shows schematically how the integration of real-time data endpoints works transparently for the modeller and the DT on the scenario description. The *vector-layer* in the user's model, here the *BicycleRentalLayer*, extends the existing *TemporalVectorLayer* type. Then, the system performs a hybrid layer integration consisting of the following steps:

- i **load meta-data:** The system reads the metadata description regarding the respective sources and retrieves the last valid data objects depending on the simulation start.
- ii **load data catalogue:** The approach assigns temporal-ordered data or source references to the respective time series according to their validity value. The data objects are added to the tail of the respective joining time series. For non-temporal data, the approach generates new model objects immediately. To this end, the temporal *vector-layer* expects a type description (in the scenario, the *BicycleRentalStation*) and initializes it with the *BicycleRentalLayer* as owner.
- iii **field replacement:** When setting one or more data stream endpoints, the configuration provides a topic or dynamic query with optional placeholders to register to. The stations do field replacement according to a fix topic pattern style in the form of *${name}*, i.e., all relevant topics are subscribed to an observed value and mapped by each *BicycleRentalStation*.
- iv **subscribe:** Entities subscribe to external real-time sources by a given query.
  - 1 Firstly, the entities may replace fields in the query against user-defined logic and provide field replacements only when model-dependent conditions apply.
  - 2 The *vector-layer* subscribes each query at the broker endpoint to receive updates matching this query. Each topic is mapped to its respective entity, thereby becoming a DTI.
- v **notify:** The system receives new data changes and adds them to the catalogue, processing with the initial data during the simulation.
- vi **create, update, remove:** During the simulation, the *vector-layer* creates new entities when new data become valid. It updates existing entities for changing data values and removes those whose lifetime of data has expired.

The indirection as a query pattern retains the possibility of attaching subsets of model objects to multiple or no real-time data sources. With the existing temporal changeability from the initial static data, this hybrid integration is transparent to the user.

6 IMPLEMENTATION

The technical design is implemented within our MARS[3] simulation system, extending the support for decoupled external data streams and arbitrary temporal system within the *vector-layer*. This extension allows monitoring the lifetime of entities.

[1]Message Queuing Telemetry Transport
[2]Constrained Application Protocol

[3]Multi-Agent Research and Simulation, www.mars-group.org

6.1 The MARS Framework

The free (GNU General Public License) and large-scale, agent-based framework MARS [16] has a significant history of applications in a variety of disciplines, e.g., climate change mitigation [4], the modeling of adaptive human behavior [22, 27], and traffic simulation [35]. MARS comprises three core concepts: agents, entities, and layers.

- *Agents* are autonomous software components that generally represent physical concepts with well-defined boundaries. In the given scenario, humans who are residents or visitors of Hamburg are modeled as agents.
- *Entities* are essentially passive agents. Without an ability to act or move autonomously, they conceptualize objects like bicycles, cars, or buses that can be utilized by agents.
- *Layers* are made up of a set of spatio-temporal data. They describe the immediate surrounding environment that agents can interact with. Besides already states vector-layer format such as GeoJSON and ESRI shape-files, MARS also supports network- and raster-layer support with ASC, GeoTiff and GraphML data. Previously only on static time-series restricted data, e.g., precipitation or temperature, can be easily incorporated into a simulation run [11].

The MARS runtime system has been implemented in *C#*, whereas models can be written in *C#* or in the MARS DSL, a domain-specific language [12]. Simulations can be executed either on local computers or in a cloud environment [35].

6.2 Registering External Sensors

External sensors are implemented in the system using pattern substitution, in which a DTI in the model registers as a subscriber to the sensor infrastructure to receive data from that source. Incoming data objects are forwarded through the temporal catalogue. When the validity period of a DTI begins, the process described in Algorithm 1 is executed for initialization.

The registration algorithm performs an update and create process for entities, starting with in an intermediate empty update set U (line 1). The process retrieves new valid data objects from the temporal catalogue T_C (line 2). Existing DTIs receive an update notification in the abstract model (e.g., the *BicycleRentalStation*), whereas new data conclude in the creation of new entities. Therefore, the algorithm *create*s an instance of the model type τ (line 7) and inserts it into the update set (line 9). The invalid join removes all remaining entities and preserves those DTIs which receive a data change (line 12) from a now invalid value to a new valid one. Therefore, the temporal catalogue T_C implements two iterators so that, if the simulation proceeds step by step, the iterators move forward to point to the beginning of the following valid or invalid data object. The linking of the data with the model entities is performed only during the simulation execution. Figure 6 shows an example iterator and simulation scope in which the valid and invalid iterator each points to respective data objects.

The *applicable* function checks whether the new valid data object d has a mapping to an existing entity object $e \in E$. If there is an object satisfying this mapping, the entity gets an *update* in which the data object is delegated. U records the extension of the lifetime scope of e by referring to the validity time of d.

Algorithm 1 Registering as Participant

```
Require:
    temporal catalogue $T_C$, entity type $\tau$,
    entity set $E$, subscription set $S$, endpoint patterns $\mathscr{P}$
1:  $U = \emptyset$
2:  $Valid \leftarrow \{d \mid d \in T_C \wedge overlapping(d_{valid\text{-}time}, sim_{time})\}$
3:  for $(d) \in Valid$ do
4:      if $\exists e : applicable(d, e)$ then
5:          $U \leftarrow U \cup \{update(e)\}$
6:      else
7:          $e_x \leftarrow create_\tau(d)$
8:          $S \leftarrow S \cup \{(e_x, topic_i) \mid \forall p_i \in \mathscr{P} : register(e_x, p_i)\}$
9:          $U \leftarrow U \cup \{e_x\}$
10:     end if
11: end for
12: $Invalid \leftarrow \{d \mid d \in T_C \wedge \neg intersecting(d_{valid\text{-}time}, sim_{time}) \wedge \neg overlapping(d_{valid\text{-}time}, sim_{time})\}$
13: for $(d) \in Invalid$ do
14:     if $\exists e : applicable(d, e) \wedge e \notin U$ then
15:         $E \leftarrow E \setminus \{e\}$ // remove
16:         $S \leftarrow S \setminus \{(e_x, topic_i) \mid \forall (e_x, topic_i) \in S : e_x = e\}$
17:     end if
18: end for
19: $E \leftarrow E \cup U$
```

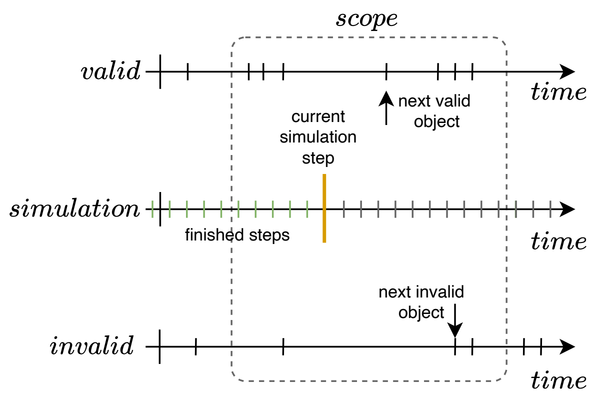

Figure 6: Valid-Invalid scope of input data objects along simulation time axis.

The *register* function (line 8) takes a pattern from the pattern set $\mathscr{P}$ and collects the placeholders using the pattern form *${name}*, replacing the field *name* with the concrete value. Each placeholder is queried against the model object e_x for the value, completing the query pattern p.

The separation of an entity from the sensor network is also performed transparently for the user by collecting all data objects from T_C that are no longer valid and checking them against applicable entities' lifetime. If the entity received no intermediate update (line 14), it needs to be removed from E (line 15). The same applies to all subscriptions S (line 16) in which the objects are dropped whose entity is similar to the currently considered one. Since the subscriptions are stored separately from the model, both the number of externally connected sensors is arbitrary, and transparency ensures the user does not need to manage the $topic_x$ himself.

7 PROOF-OF-CONCEPT: HAMBURG BICYCLE RENTAL STATIONS

Given the DT of Hamburg (Section 4) and the data integration architecture (Section 5) as well as its implementation (Section 6), a scenario was prepared to serve as a proof-of-concept for the mechanism by which deviations accrued during a simulation can be corrected[4]. The mechanism is applied to passive DTIs of real-world RS in the DT – each of which is modeled as a *BicycleRentalStation* – which agents can interact with by querying the *BicycleRentalLayer*. The proof-of-concept strives to use the data integration mechanism to adjust, if necessary, the number of *RentalBicycle*s per *BicycleRentalStation* at the end of a simulation using real-time and historical data provided by the city's sensor network. In particular, the Hamburg SensorThings API platform[5] is integrated – though the set of sensors providing real data is presently still very limited.

7.1 Scenario

In the district of Hamburg-Harburg (the simulation area, hereafter HH-Harburg), there are 14 RS, making up 5,6% of all RS in Hamburg as of 2020. Between 4:00 pm and 6:00 pm, there are 1,045 trips made by bicycle in HH-Harburg, 20% of which (roughly 320) are made using a bicycle from a RS. On a monthly scale, this number is further reduced by 50%: People who use a bicycle from a RS do so less than once a month (data from mobility report MiD 2017 [26]). Therefore, at most, 160 trips using a bicycle from a RS are expected to occur during the simulation window. There are three trip categories – local, outbound, and inbound [34] – among which the number of trips is divided evenly. Given the upper bound of 160, the extent of under-usage of bicycles from RS is to be determined.

A configuration of spawning *CycleTraveler* agents with reasonable goals is predefined using MiD 2017. The analyzed period was two hours on Monday, December 14th, 2020. The scenario is configured to start at 4:00 pm (t_{start}) with real-time sensor data for the 14 RS located in HH-Harburg. During each simulation time step (Δt = 15 minutes), the simulation outputs the current number of bicycles per RS. Simultaneously, corresponding real-time data are documented. At 6:00 pm, all simulated data are corrected to the real-time data.

7.2 Results

Figure 7 depicts the deviation between the ground truth and the simulated value of available bikes at 14 RSs. At the end of the – parameterizable – time interval the simulation states were corrected to the real numbers. The diagram also illustrates that the variance of the simulated values from the real world increases over simulation time, and the synchronisation with the real-time data shows an apparent corrective effect.

As expected, the data generally overestimate the use of bicycles from RS on this particular winter day. Although we are aware that temperature and precipitation have a significant impact on cycling [14], the simulation did not consider weather data. In total, after two hours of simulation time, 11 bicycles were removed from the simulation at specific RS and 13 bicycles were added at other RS (for more details, see Table 1 in Appendix: Raw Data).

[4]https://git.haw-hamburg.de/mars/model-deployments
[5]https://iot.hamburg.de/

8 DISCUSSION

The bicycle rental station extension to the existing traffic model of Hamburg reflects the implementation of a passive DTI. RS sensors and the software model are unidirectionally linked, i.e., data flow from the sensors to the entities or agents only. An active DTI implementation would require that the real-world system can be operated by the model, whereas an interactive DTI would be based on a bidirectional interface between the virtual and the real world. It becomes apparent that especially active and interactive DTIs demand a special focus to safety and security issues.

The simulation scenario displays the potential of temporal data for reducing the corridor of uncertainty at multiple synchronization points before predicting future states. The data trajectories seen in Figure 7 indicate that the extent of the corridor may grow over the course of a simulation. The approach of coupling *BicycleRentalStation*s in the model with real-world sensors and getting updates from their physical counterparts led to corrected model states and simulation outputs. Furthermore, it has shown how travel planning of agents became affected (e.g., by adaptive replanning).

This integration comes with problems and in particular addresses – aside from technical aspects such as missing service availability – the mapping between model components and real-world objects. The problems concern the following:

- Model objects may be initialized from different sources (files, databases) and have to be coupled with sensor data.
- Data preparation includes correcting, normalizing, and identifying duplicates and outliers from integrated sources.
- Change events from sensors can occur at missing time points, unknown time points, or intervals that do not overlap with the simulation time. These need to be correlated or omitted during processing. Reasons for this can be the latency of incoming data messages.
- The life span of data objects coming from multiple sources is arbitrary and needs to be ordered chronologically. Such an ordering has to prioritize some values over others when validity periods are overlapping or intersecting.

One major challenge in creating a general-purpose integration was to join the real-world data objects with the model parts by marking data as valid or invalid and populating them accordingly.

We decided to increase or decrease the number of available *RentalBicycle*s at the corresponding *BicycleRentalStation*s. The model did not prevent *CycleTraveler*s from using non-corrected station states so that bicycles that are already rented remain unaffected. Due to missing information about current rentals, we consider this approach a mixture of soft and hard correction. A hard correction would correct all model parts that are dependent on the data inputs (i.e., also remove *RentalBicycles* that are on the road at the time of correction). Having implemented a relatively hard correction mechanism enabled us to obtain a quantitative measure of the corridor of uncertainty over simulation time. This measure has the potential to serve as a validation tool during model development and simulation data. Generally, all synchronization strategies aim,

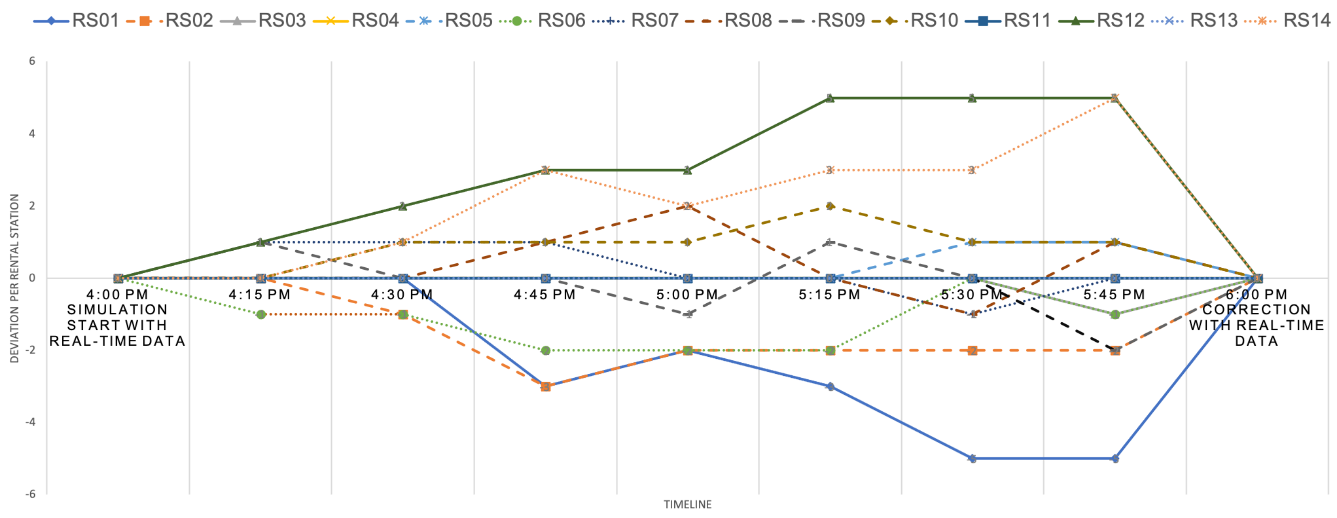

Figure 7: Deviation of the simulated number of available bicycles from real-world data for selected bike sharing stations.

at least in part, to increase or maintain the validity of the simulation. Other strategies may prove more useful depending on the domain or scenario at hand. For example, a softer correction could consist of additionally computing the average between the simulated and the next valid value. Alternatively, a tougher condition would be to select the maximum/minimum as well as simply assigning them and removing all dependent objects. An analysis of different update strategies and their impacts on a given scenario is a promising area for future research and experimentation.

The usage of synchronization points was highlighted in the simulation scenario. All update messages are stored in the model objects until a synchronization event is called that applies the updated information to the model object (in our scenario: the creation of a *BicycleRentalStation* or the removal of *RentalBicycles* from existing *BicycleRentalStations*). Another mode could be an immediate synchronization of updates to model objects when new data become valid. This case distinction facilitates multiple use cases for incorporating real-time data for DTs:

- A DT can start a simulation in the past and surpass the current time in the physical world. While simulating, sensor updates are still added to the temporal catalogue and the synchronization is applied when the simulation time intersects the current real time. Therefore, the model is populated with the newest available data when the simulation becomes predictive.
- Instead of predicting the future when the current real time is reached, the execution of the simulation could slow down to advance at the same rate as the real time and synchronize the DT with its physical counterparts. The system continues to forward real-world updates. This type of execution allows for focusing on aspects in situations that cannot be concluded with existing data or infrastructure.

The overarching integration with its temporal change is basically domain-independent and transparent, but there are many aspects that still have to be realized by the domain model. This includes the integration of new data in the entities with its effects on the agents.

9 CONCLUSION

We proposed a conceptual and technical framework for coupling an existing traffic model of Hamburg, Germany, with the city's IoT network, realizing a DT according to the definition given in Section 3. The streaming of real-time data into the ABM during runtime allowed for a continuous 'correction' of agent states, i.e., creating an agent-related, local rollback mechanism.

Intrinsically, the risk that simulative predictions deviate from the real-world situation increases with the duration of a simulation (particularly, when simulating into the future). The left diagram in Figure 8 depicts this idea. It should be noted that the emerging 'corridor of uncertainty', i.e., the gap between the real world and the simulation state, does not necessarily develop linearly or symmetrically. Thus, the diagram illustrate merely one possible trajectory for illustrative purposes.

The right diagram in Figure 8 reflects the observed behavior from this study. In the proof-of-concept described in Section 7, the number of rental bikes available to the agents were corrected during simulation time. Both the simulation runtime system and the software agents need to address such corrections in their logic. It becomes apparent that this adaptive dynamic is generally neither possible nor sensible under any circumstances. However, referring to the analyses from Clay et al. [9] and Kieu et al. [18] cited above, we suggest that the study's findings as a first basic step for dynamically incorporating real-time data into ABMs.

We further proposed a formal differentiation between ABMs and DTs, hoping that this can contribute to a more precise distinction between and usage of the two concepts. The growing number of fields where the DT concept is applied will show how robust this formal differentiation actually is. In this context, it needs to be evaluated if this paradigm will lead to the indistinguishability of the real system and corresponding models as assumed by Batty [3].

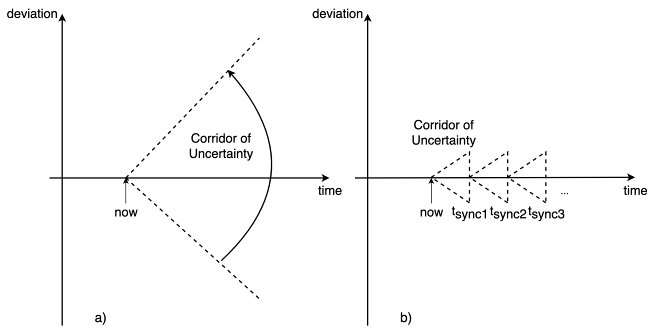

Figure 8: Schematic 'corridor of uncertainty' due to the deviation between the simulated and the real world. a) normal simulation runs; b) simulations executed by a DT as suggested in this study.

Additionally, three different levels of complexity were suggested for DTIs in this study. The proof-of-concept presented here only reflects one of them, the passive DTI. The active DTI is not covered since the underlying principles are entirely different. The interactive DTI is actually a union of the two former types.

We find that ABMs are well-suited for implementing DTIs and DTs due to the notion of autonomy inherent to the concept of agents. With a growing number of IoT sensor networks accessible to ABMs, scalability and robustness will become key features. It is assumed that DTs will play a vital role in the development, maintenance, and management of Smart Cities in the near future.

ACKNOWLEDGMENTS

This research was supported in part by the City of Hamburg (SmartOpenHamburg project).

REFERENCES

[1] Heiko Aydt, Steven John Turner, Wentong Cai, and Malcolm Yoke Hean Low. 2008. Symbiotic Simulation Systems: An Extended Definition Motivated by Symbiosis in Biology. In *2008 22nd Workshop on Principles of Advanced and Distributed Simulation.* 109–116. https://doi.org/10.1109/PADS.2008.17

[2] Michael Batty. 2018. Digital twins. *Environment and Planning B: Urban Analytics and City Science* 45, 5 (2018), 817–820. https://doi.org/10.1177/2399808318796416

[3] Michael Batty. 2019. A map is not the territory, or is it? *Environment and Planning B: Urban Analytics and City Science* 46, 4 (2019), 599–602. https://doi.org/10.1177/2399808319850652

[4] Christian Berger, Mari Bieri, Karen Bradshaw, Christian Brümmer, Thomas Clemen, Thomas Hickler, Werner Leo Kutsch, Ulfia A. Lenfers, Carola Martens, Guy F. Midgley, Kanisios Mukwashi, Victor Odipo, Simon Scheiter, Christiane Schmullius, Jussi Baade, Justin C.O. du Toit, Robert J. Scholes, Izak P.J. Smit, Nicola Stevens, and Wayne Twine. 2019. Linking scales and disciplines: an interdisciplinary cross-scale approach to supporting climate-relevant ecosystem management. *Climatic Change* 156, 1 (2019), 139–150. https://doi.org/10.1007/s10584-019-02544-0

[5] Carsten Bormann, Angelo P Castellani, and Zach Shelby. 2012. Coap: An application protocol for billions of tiny internet nodes. *IEEE Internet Computing* 16, 2 (2012), 62–67.

[6] Sanjay Chawla and Pei Sun. 2006. SLOM: A New Measure for Local Spatial Outliers. *Knowl. Inf. Syst.* 9, 4 (2006), 412–429. https://doi.org/10.1007/s10115-005-0200-2

[7] Yao-Yi Chiang, Bo Wu, Akshay Anand, Ketan Akade, and Craig A. Knoblock. 2014. A System for Efficient Cleaning and Transformation of Geospatial Data Attributes. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* (Dallas, Texas). ACM, 577–580. https://doi.org/10.1145/2666310.2666373

[8] Claudio Cioffi-Revilla, J. Daniel Rogers, and Atesmachew Hailegiorgis. 2011. Geographic information systems and spatial agent-based model simulations for sustainable development. *ACM Transactions on Intelligent Systems and Technology* 3, 1 (2011), 1–11. https://doi.org/10.1145/2036264.2036274

[9] Robert Clay, Le Minh Kieu, Jonathan A. Ward, Alison Heppenstall, and Nick Malleson. 2020. Towards Real-Time Crowd Simulation Under Uncertainty Using an Agent-Based Model and an Unscented Kalman Filter. In *Advances in Practical Applications of Agents, Multi-Agent Systems, and Trustworthiness. The PAAMS Collection*, Vol. 12092 LNAI. 68–79. https://doi.org/10.1007/978-3-030-49778-1_6

[10] Venkatesh Ganti and Anish Das Sarma. 2013. *Data Cleaning: A Practical Perspective.* Morgan & Claypool Publishers.

[11] Daniel Glake, Norbert Ritter, and Thomas Clemen. 2020. Utilizing Spatio-Temporal Data in Multi-Agent Simulation. In *2020 Winter Simulation Conference (WSC)*, K.-H. Bae, B. Feng, S. Kim, S. Lazarova-Molnar, Z. Zheng, T. Roeder, and R. Thiesing (Ed.). Institute of Electrical and Electronics Engineers (IEEE), 242–253. https://doi.org/10.1109/WSC48552.2020.9384124

[12] Daniel Glake, Julius Weyl, Carolin Dohmen, Christian Hüning, and Thomas Clemen. 2017. Modeling through Model Transformation with MARS 2.0. In *Proceedings of the Agent-Directed Simulation Symposium* (Virginia Beach, Virginia) *(ADS '17)*. Society for Computer Simulation International, San Diego, CA, USA, Article 2, 12 pages.

[13] Manish Gupta, Jing Gao, Charu C. Aggarwal, and Jiawei Han. 2014. Outlier Detection for Temporal Data: A Survey. *IEEE Trans. Knowl. Data Eng.* 26, 9 (2014), 2250–2267. https://doi.org/10.1109/TKDE.2013.184

[14] Steve Hankey, Wenwen Zhang, Huyen T.K. Le, Perry Hystad, and Peter James. 2021. Predicting bicycling and walking traffic using street view imagery and destination data. *Transportation Research Part D: Transport and Environment* 90, December 2020 (2021), 102651. https://doi.org/10.1016/j.trd.2020.102651

[15] Andreas Horni, Kai Nagel, and Kay W Axhausen (Eds.). 2016. *The Multi-Agent Transport Simulation Title of Book : The Multi-Agent Transport Simulation MATSim.* Ubiquity Press.

[16] Christian Hüning, Mitja Adebahr, Thomas Thiel-Clemen, Jan Dalski, Ulfia Lenfers, and Lukas Grundmann. 2016. Modeling & Simulation as a Service with the Massive Multi-Agent System MARS. In *Proceedings of the Agent-Directed Simulation Symposium* (Pasadena, California) *(ADS '16)*. Society for Computer Simulation International, San Diego, CA, USA, Article 1, 8 pages.

[17] Ihab F. Ilyas and Xu Chu. 2019. *Data Cleaning.* ACM. https://doi.org/10.1145/3310205

[18] Le Minh Kieu, Nicolas Malleson, and Alison Heppenstall. 2020. Dealing With Uncertainty in Agent-Based Models for Short-Term Predictions. *Royal Society open science* 7, 1 (2020). https://doi.org/10.1098/rsos.191074

[19] Yufeng Kou and Chang-Tien Lu. 2017. *Outlier Detection, Spatial.* Springer International Publishing, Cham, 1539–1546. https://doi.org/10.1007/978-3-319-17885-1_945

[20] Ioannis K. Koumarelas, Lan Jiang, and Felix Naumann. 2020. Data Preparation for Duplicate Detection. *ACM J. Data Inf. Qual.* 12, 3 (2020), 15:1–15:24. https://dl.acm.org/doi/10.1145/3377878

[21] Petar Krivic, Pavle Skocir, and Mario Kusek. 2018. Agent-based approach for energy-efficient IoT services discovery and management. *Smart Innovation, Systems and Technologies* 96, January (2018), 57–66. https://doi.org/10.1007/978-3-319-92031-3_6

[22] Ulfia A. Lenfers, Julius Weyl, and Thomas Clemen. 2018. Firewood collection in South Africa: Adaptive behavior in social-ecological models. *Land* 7, 3 (2018), 97. https://doi.org/10.3390/land7030097

[23] Steve Liang, Chih-Yuan Huang, and Tania Khalafbeigi. 2016. *OGC SensorThings API Part 1: Sensing, Version 1.0.* Retrieved April 7, 2021 from http://www.opengis.net/doc/is/sensorthings/1.0

[24] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y. P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wiessner. 2018. Microscopic Traffic Simulation using SUMO. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC).* 2575–2582. https://doi.org/10.1109/ITSC.2018.8569938

[25] Ryo Neyama, Sylvain Lefebvre, Masanori Itoh, Yuji Yazawa, Akira Yoshioka, Jun Koreishi, Akihisa Yokoyama, Masahiro Tanaka, and Hiroko Okuyama. 2020. Simulating Vehicular IoT Applications by Combining a Multi-agent System and Big Data. In *Engineering Multi-Agent Systems*, Cristina Baroglio, Jomi F. Hubner, and Michael Winikoff (Eds.). Springer International Publishing, Cham, 119–128.

[26] Claudia Nobis and Tobias Kuhnimhof. 2018. *Mobilität in Deutschland – MiD Ergebnisbericht.* Technical Report. Bundesministers für Verkehr und digitale Infrastruktur, Bonn/Berlin, Germany. 1–133 pages. http://www.mobilitaet-in-deutschland.de/pdf/MiD2017_Ergebnisbericht.pdf

[27] Devotha G Nyambo, Edith T Luhanga, Zaipuna O Yonah, Fidalis D N Mujibi, and Thomas Clemen. 2020. Leveraging peer-to-peer farmer learning to facilitate better strategies in smallholder dairy husbandry. *Adaptive Behavior* (2020). https://doi.org/10.1177/1059712320971369

[28] B. S. Onggo, N. Mustafee, A. Smart, A. A. Juan, and O. Molloy. 2018. SYMBIOTIC SIMULATION SYSTEM: HYBRID SYSTEMS MODEL MEETS BIG DATA ANALYTICS. In *2018 Winter Simulation Conference (WSC).* 1358–1369. https://doi.org/10.1109/WSC.2018.8632407

Table 1: Results of simulated data versus real-time data

Rental Station (RS)	Bahnhof Harburg / Moorstraße	Denickestraße / TUHH	Eißendorfer Pferdeweg / Asklepios-Klinik Harburg	Göhlbachtal / Berufliche Schule Harburg	Gotthelfweg / Außenmühlen teich	Harburger Ring / Neue Straße	Herbert-Wehner-Platz / Großer Schippsee	Kanalplatz / Harburger Schlossstraße	Maretstraße / Baererstraße	Neuländer Kamp / Eurofins	Reeseberg / Anzengruber straße	S Harburg Rathaus / Deichhausweg	S Heimfeld / Alter Postweg	Schellerdamm / Hausnummer 22
Abbreviation	RS01	RS02	RS03	RS04	RS05	RS06	RS07	RS08	RS09	RS10	RS11	RS12	RS13	RS14
Characteristic	railway station	university	hospital	school	x	x	x	x	x	company	x	public transport / S-Bahn station	public transport / S-Bahn station	x
Real-time data	**Data from sensor network**													
4:00 PM	13	10	4	1	3	9	9	8	0	21	2	11	0	25
4:15 PM	13	10	4	1	3	9	10	8	1	21	2	11	0	25
4:30 PM	13	10	4	1	3	9	10	8	1	21	2	11	0	25
4:45 PM	13	8	4	1	3	9	10	8	1	21	2	11	0	26
5:00 PM	13	9	4	1	3	10	9	8	1	21	2	11	0	26
5:15 PM	13	9	4	1	3	10	9	8	2	21	2	11	0	26
5:30 PM	13	8	4	1	4	11	8	8	2	20	2	11	0	26
5:45 PM	13	9	3	1	4	10	9	9	1	20	2	11	0	26
6:00 PM	13	9	3	1	4	10	9	9	1	20	2	11	0	26
Simulation data	**ABM simulated data (DT)**													
4:00 PM	**Simulation starts with real-time data**													
4:15 PM	13	10	4	1	3	10	9	8	0	21	2	10	0	25
4:30 PM	13	11	4	1	3	10	9	8	1	20	2	9	0	24
4:45 PM	16	11	4	1	3	11	9	7	1	20	2	8	0	23
5:00 PM	15	11	4	1	3	12	9	6	2	20	2	8	0	24
5:15 PM	16	11	4	1	3	12	9	8	1	19	2	6	0	23
5:30 PM	18	10	4	1	3	11	9	9	2	19	2	6	0	23
5:45 PM	18	11	4	1	3	11	9	8	3	19	2	6	0	21
6:00 PM	**Simulation ends by correction with real-time data**													

[29] Pablo Pico-Valencia and Juan A. Holgado-Terriza. 2018. Agentification of the Internet of Things: A systematic literature review. *International Journal of Distributed Sensor Networks* 14, 10 (2018). https://doi.org/10.1177/1550147718805945

[30] Dorian Pyle. 1999. *Data Preparation for Data Mining.* Morgan Kaufmann.

[31] Adil Rasheed, Omer San, and Trond Kvamsdal. 2019. Digital Twin: Values, Challenges and Enablers. arXiv:1910.01719 [eess.SP]

[32] Eduardo F.Z. Santana, Ana P. Chaves, Fabio Kon, Marco A. Gerosa, and Dejan J. Milojicic. 2017. Software Platforms for Smart Cities: Concepts, Requirements, Challenges, and a Unified Reference Architecture. *ACM Comput. Surv. Article* 50, 78, Article 78 (Nov. 2017), 37 pages. https://doi.org/10.1145/3124391

[33] Claudio Savaglio, Maria Ganzha, Marcin Paprzycki, Costin Bădică, Mirjana Ivanović, and Giancarlo Fortino. 2020. Agent-based Internet of Things: State-of-the-art and research challenges. *Future Generation Computer Systems* 102 (2020), 1038–1053. https://doi.org/10.1016/j.future.2019.09.016

[34] Statistical Office of Hamburg and Schleswig-Holstein. 2018. *Employed persons in Hamburg 2018 (in German).* Retrieved November 26, 2020 from https://www.statistik-nord.de/zahlen-fakten/erwerbstaetigkeit-verdienste-arbeitskosten/erwerbstaetigkeit/dokumentenansicht/erwerbstaetige-in-hamburg-2018-60940

[35] Julius Weyl, Ulfia A. Lenfers, Thomas Clemen, Daniel Glake, Fabian Panse, and Norbert Ritter. 2019. Large-Scale Traffic Simulation for Smart City Planning with Mars. In *Proceedings of the 2019 Summer Simulation Conference* (Berlin, Germany) *(SummerSim '19)*. Article 2, 12 pages.

[36] Rainer Wiedemann. 1974. *Simulation des Straßenverkehrsflusses.* Ph.D. Dissertation. Karlsruher Institut für Technologie (KIT), Karlsruhe.

[37] Clare Wildfire. 2018. *How can we spearhead city-scale digital twins?* Retrieved April 7, 2021 from http://www.infrastructure-intelligence.com/article/may-2018/how-can-we-spearhead-city-scale-digital-twins

[38] Aoqian Zhang, Shaoxu Song, Jianmin Wang, and Philip S. Yu. 2017. Time Series Data Cleaning: From Anomaly Detection to Anomaly Repairing. *Proc. VLDB Endow.* 10, 10 (2017), 1046–1057. https://doi.org/10.14778/3115404.3115410

[39] Song Zheng, Qi Zhang, Rong Zheng, Bi Qin Huang, Yi Lin Song, and Xin Chu Chen. 2017. Combining a multi-agent system and communication middleware for smart home control: A universal control platform architecture. *Sensors* 17, 9 (2017). https://doi.org/10.3390/s17092135

A APPENDIX: RAW DATA

Table 1 lists the names, acronyms, and some characteristics of the 14 RS in HH-Harburg whose DTIs were the subject of the proof-of-concept presented in Section 7. Every 15 minutes, data from the IoT network were read and documented (numbers in black). The simulation began at 4:00 pm with real-time data and at every $\Delta t = 15$, the number of bicycles available at the RS in the simulation was counted and aggregated per RS (numbers in red). At 6:00 pm, the real-world counts were used to adjust the bicycle count at each *BicycleRentalStation* in the simulation.