© 2021 by the authors; licensee RonPub, Lübeck, Germany. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution license (http://creativecommons.org/licenses/by/4.0/).

Open Access

Open Journal of Internet of Things (OJIOT)
Volume 7, Issue 1, 2021
http://www.ronpub.com/ojiot
ISSN 2364-7108

Realizing the Digital Twin Transition for Smart Cities

Jonathan Fürst, Bin Cheng, Benjamin Hebgen

NEC Laboratories Europe GmbH, Kurfürsten-Anlage 36, Heidelberg, Germany,
{jonathan.fuerst, bin.cheng, benjamin.hebgen}@neclab.eu

Abstract

The digital twin transition for cities is expected to improve, among others, living quality, carbon footprint and generate new business opportunities across different organizations. However, as cities consist of many separate entities that are in close and frequent interaction with each other, it is not possible to simply apply digital twin concepts from the engineering and manufacturing domains in a silo-ed fashion for each entity. In this paper, we distill the requirements and challenges to develop digital twins for smart cities based on a typical smart city use case. We follow with a first systematic approach to address them in a data-driven fashion to realize the digital twin transition for cities.

Type of Paper and Keywords

Visionary paper: *Digital Twin, Urban Digital Twin, Machine Learning, IoT, Edge Computing, Knowledge Infusion, Smart City*

1 Introduction

Digital twins have seen increasing attention beyond their originally intended application in the context of aerospace engineering and quickly been picked up by the manufacturing domain as part of the so-called Industry 4.0 revolution [22]. Recently, the concept is being further adapted to diverse domains such as Health [3] or applied in Smart Cities for collaborative urban planning [7]. While the original concept has been focused mainly on a—potentially very precise—simulation of physical objects or systems together with updates through sensor data to make predictions about the underlying object (e.g., fatigue prediction [1]), the adaption of the digital twin concept to the Internet of Things and in particular to Smart Cities brings new challenges and opportunities. Opposed to previous application areas of digital twins, cities are complex cyber-physical human ecosystems that consists of many different organizations. The underlying data is highly distributed, of high variety and there exists no single data ownership, making the digital twin transition challenging.

This paper is accepted at the *International Workshop on Very Large Internet of Things (VLIoT 2021)* in conjunction with the VLDB 2021 conference in Copenhagen, Denmark. The proceedings of VLIoT@VLDB 2021 are published in the Open Journal of Internet of Things (OJIOT) as special issue.

In order to realize the digital transition, for cities, we need to move from domain specific digital twins, to a network of connected digital twins across different sectors. In order to scale to a whole city, we need to move from their manual construction to a (semi-) automated, data-driven generation, replacing sophisticated physical simulations with data-driven AI techniques.

The underlying foundation for this digital twin transition is data. Data needs to be harmonized across different data silos in order to enable a distributed knowledge graph across data silos, enabling the virtualization of physical city processes, which happen across many organizations.

In this paper, we present challenges and a first systematic approach to enable digital twins for the Internet of Things (IoT) in the Smart City domain. Our approach is data driven, mapping and adapting existing concepts and methods from the database community to realize Digital Twins for Smart Cities. In summary our findings and contributions are the following:

- Data to build digital twins in smart cities is usually created at the edges of the network, often by various sensors that are connected via low-bandwidth interfaces. We thus need to move from traditional Big Data processing in the cloud (Spark, Hadoop etc.) to data processing at the edges, both for performance reasons and for privacy reasons. For our digital twin platform, we utilize FogFlow, an open-source and standards based cloud-edge platform that supports serverless computing [5, 4].
- Data in a city is owned by different entities. For example, each city department and many businesses collect and store their own data in a silo-ed fashion. This is in strong contrast to a city being in the real world a complex ecosystem in which many entities are in constant interaction with each other and many city goals require the cooperation of different organizations. Thus, to enable the promises of the digital twin transition, data cannot be created and stored in a centralized fashion, but needs to be distributed/federated across different owners of data silos, giving data owners fine-grained access control. Knowledge graphs have proven successful in large scale data representations (e.g., Google Knowledge Graph [9], Amazon Product Graph [10]) and are a natural choice to represent the network of humans, objects and their interactions in a city. In our platform, we achieve a federated knowledge graph through the linked data standard and protocol NGSI-LD [14] and an implementation of Scorpio, an NGSI-LD broker supporting federation across multiple broker instances [21].
- Data integration has been addressed by many works of the database community [12], moving recently to machine learning based techniques [11]. The digital twin transition for cities also poses a data integration problem as data is generated by multiple different organizations, often in an ad hoc fashion. However, compared to many other data integration problems, data cannot just be integrated centrally as there exist many legal and data confidentiality issues (e.g., the recent European GDPR law). In our method, we instead apply a privacy preserving linking/mapping on schema level, without revealing instance data (e.g., the actual sensor data is not revealed).
- Last, we recognize the need for a paradigm shift from purely simulation based digital twins to increasingly data-driven digital twins in order to scale the digital twin concept to a city level. In a city, digital twin behavior, such as prediction tasks can often be implemented using machine learning techniques. As domain experts are often not data scientists and the manual creation of labeled data poses a bottleneck for IoT, we propose a weak-supervision interface for domain experts to infuse knowledge [18]. Interactive techniques such as active learning can further be used to enable ML for the digital twin long tail.

In the following, we first motivate digital twins with a smart city use case and derive requirements. We identify open challenges and then present our systematic approach for a twin transition in an open ecosystem.

2 Digital Twins across Silos

We start with a concrete use case on smart cities to explain how digital twins are expected to work from the user's perspective and then introduce our definition of digital twins for the purpose of enabling efficient data sharing and utilization across data silos. Based on that, we analyze the requirements of IoT platforms to realize the transition from purely data-oriented sharing to twin-oriented sharing.

2.1 Use Case

In a physical environment like smart city or smart campus many things like parking lots, cameras, or vehicles are already connected, but very often they are isolated in different data silos and it is hard for them to share and exchange contextual information in time. These silos are now becoming the biggest barrier for us to achieve the real smartness of IoT devices. Creating the digital twins of those connected devices at a virtual twin layer across data silos could be a promising solution to break the data silos and enable easy and efficient data sharing and utilization.

Take smart parking as an example. As illustrated in Figure 1, in a smart city there exists usually many parking sites, including public parking sites managed by the city (Silo-A) and private parking sites managed by a private company (Silo-B). When someone drives to the city center for shopping with an advanced

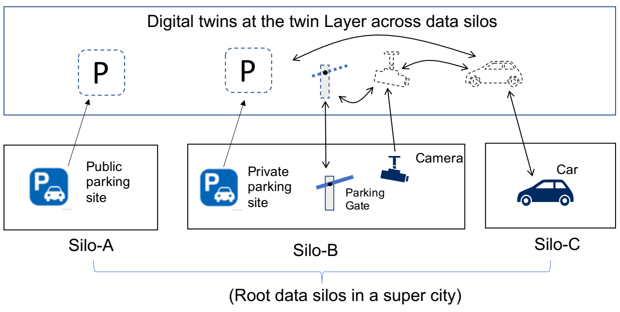

Digital twins at the twin Layer across data silos
P
P
Public parking site
Private parking site
Parking Gate
Camera
Car
Silo-A
Silo-B
Silo-C
(Root data silos in a super city)

Figure 1: Digital twins across data silos

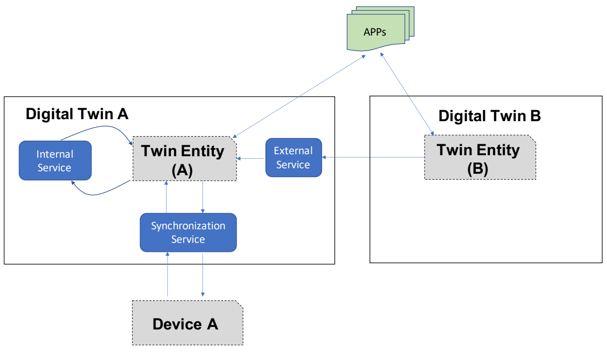


Figure 2: Elements of a digital twin

vehicle, such as an electric car, she/he likes to receive a timely recommendation on where to park. In this case, the driver might expect that the twin of the car communicates with the twins of all parking lots in the destination area, together with the road traffic information on the way, to figure out the best parking lot to park and keep the driver informed timely. When the car arrives at the entry point of the parking site, a connected camera can capture the plate number of the car and then trigger the parking gate to open automatically. Also, the car can guide the driver to the right parking lot that has been reserved by the car on behalf of the driver. With the help of the digital twins that live at the twin layer to link those isolated data silos, the entire process could be carried out efficiently and seamlessly with minimal interference from the car driver.

2.2 Concept of Digital Twins

To enable efficient data sharing and exchange across data silos, we propose to abstract the concept of a digital twin as illustrated in Figure 2. A digital twin consists of a standard based data presentation plus a set of atomic services. Its data presentation is presented by a twin entity with the semantically annotated data schema. Each twin can have a set of atomic services around the data entity to update, enrich, and maintain its data presentation, including:

- **Synchronization service** to enable the bi-directional data transformation and synchronization between the twin entity of a twin and its corresponding physical object. The service is responsible for dealing with the heterogeneity and interoperability of communication protocol and data model on both sides. E.g., a building twin needs to be updated based on sensor values from the Building Management System (BMS), handling various protocols such as ModBus or BACnet and potentially inconsistent data models.

- **Internal service** to take a set of existing properties as inputs to perform some data analytics and then use the generated analytics result to create or update a property for the same twin entity. E.g., an internal service might use signal strength data from WiFi or Bluetooth sources to estimate building occupancy [28].

- **External service** to take the input data from other twin entities to perform some data analytics and then use the generated analytics result to create or update a new property for the local twin entity. E.g., the building twin might query the mobility twins of the building users to predict the arrival times of people in order to then actuate the HVAC system accordingly.

With our design, the data sharing and exchange between digital twins or with the upper layer twin applications can be achieved via a unified and standardized interface of data entities. All atomic services could stay behind the actual data usage of twin entities to make sure that the twin entity can always reflect the latest view of a digital twin with the required information.

2.3 Requirements

We analyze the requirements of creating the designed digital twins for cross-silo data sharing. Here we exclude the security related requirements, which are out of the scope of this paper. In terms of functional requirements, we aim to cover the following aspects:

1. Twin Construction: to create the twin entity out of heterogeneous data sources as the data presentation step of a digital twin.

2. Creation of atomic service: the major capability of a digital twin will be augmented by its atomic services. Therefore, there is an essential need to

program the logic of atomic services via some high level programming model. In some case, some atomic services rely on some trained AI models to make decisions according to its contextual information. How to seamlessly create AI models is also an important part of service creation.

3. Orchestration of atomic service: an execution environment is required to instantiate those atomic services and run them seamless over a cloud and edge environment.
4. Interoperability and interaction between digital twins: to make sure different twins from different silos can interact with each other via common data models and communication protocols.

In addition, we also take into account the following non-functional requirements:

1. Automation: we need to automate the entire processes of twin creation, service creation, and service orchestration as much as possible.
2. Scalability: the twin layer across data silos must be scalable to support a large number of silos.

3 Challenges

We recognize the following challenges:

[C1] *Data integration only on schema level.* When data is federated across silos in multiple organizations, confidentiality and legal regulations (e.g., the European data privacy GDPR law) are an obstacle for traditional centralized data integration approaches. Therefore, practically data integration needs to occur in two steps: (1) Integration on a schema level, without sharing and utilizing confidential information about the underlying instance data and (2), potentially, the sharing and integration of instance data, depending on the identified opportunities and legal grounding. Existing data integration solutions usually are usually only tailored for (2).

[C2] *Traditionally data driven ML approaches do not generalize well across IoT deployments.* As has been shown [18], traditionally supervised ML models often do not provide robust performance in face of domain shifts, without re-training on the specific data distribution. Re-training requires new labeled data, which is too expensive to obtain, as it often requires a human to manually annotate events in a data stream by observing the events in the real world. All this makes the creation of ML based atomic services for digital twins cumbersome.

[C3] *Data is produced at edges.* For performance, latency and potentially privacy reasons, there is a need for edge computing. Computation needs to move closer to the data instead moving the data closer to computation.

4 Our Systematic Approach

Our digital twin platform consists of four layers: (1) A device layer, which connects various IoT infrastructures, such as sensors and their used protocols; (2) A cloud-edge runtime, which enables data processing flows from the edge to the digital twin layer; (3) The digital twin layer, which consists of NGSI-LD modeled knowledge graphs accessible through a context broker and (4) The federation layer, which enables data exchange between different brokers at different data silos, facilitating schema matching and federated learning. Domain experts are able to introduce their knowledge in form of weak supervision signals and active learning input to train machine learning models without exhaustive manual labeling effort. These ML models are used to enable data extraction from the edge to the digital twin, implement digital twin behavior (e.g., parking place prediction) and also can support the schema matching across different data silos.

4.1 Knowledge Graphs with NGSI-LD and Scorpio

Scorpio [21] is an open source context broker implementing the NGSI-LD API as specified by the ETSI Industry Specification Group [14]. The NGSI-LD API enables the management, access and discovery of context information. Context information is modeled in a graph structure, which consists of entities (e.g., a building) and their properties (e.g., address and geographic location) and relationships to other entities (e.g., owner and users). Thus Scorpio enables applications and services to request context information—what they need, when they need it and how they need it. This allows the modeling of digital twins of real world entities (see Figure 4).

NGSI-LD provides a wide set of methods to interact with context information and thus the digital twin. CRUD (Create Read Update Delete) methods for entities are available as well as a more sophisticated query system and a subscription system. The query system includes various filtering and scoping (e.g. geographic) capabilities which enable users or services

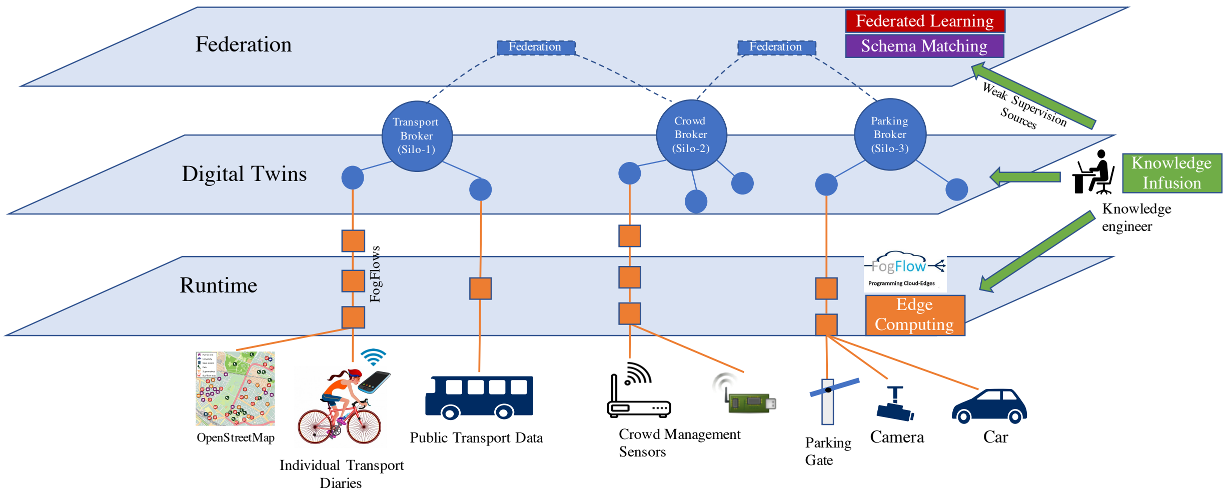


Figure 3: System Overview. Digital twins are constructed from IoT data sources by FogFlow processing flows and federated via Scorpio NGSI-LD Federation Broker. Users can infuse domain knowledge as weak supervision sources.

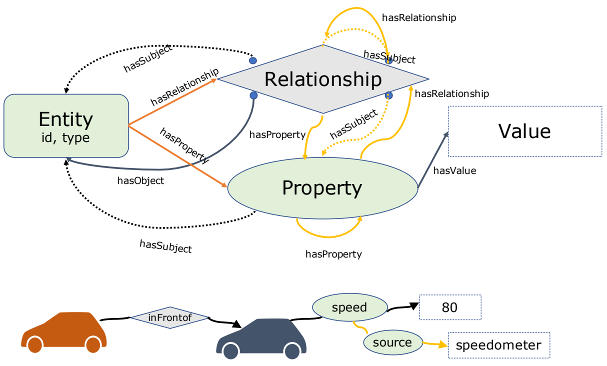


Figure 4: NGSI-LD Entity Model

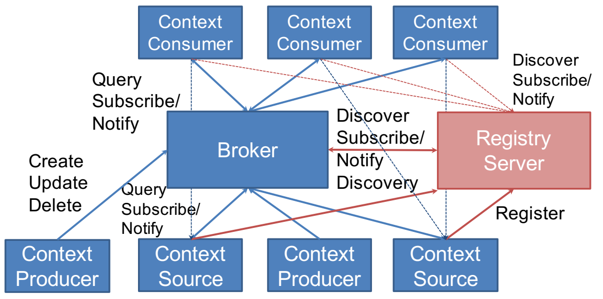


Figure 5: NGSI-LD Broker Setup with Registry

to find specific entities, e.g. a query could return all entities which have a relationship to another entity. Subscriptions provide an asynchronous way to receive context information when they change.

NGSI-LD defines two types of data sources for context data. Context Producers are actively pushing context information into the broker. Context Sources, which provide a subset of the NGSI-LD API, allowing them directly to be queried and subscribed to. In order to discover these Context Sources via a broker NGSI-LD defines a registry interface where Context Sources register their endpoint and their provided entities. Scorpio uses the registry directly in every query and subscription to combine all available variants of an entity into a single result entity (see Figure 5).

This allows Scorpio to provide a digital twin from various data sources provided as an entity. Looking at the real world we very often come across the situation that data sources are spread across different departments, companies or service providers which all need to have their own data sovereignty and hence cannot all report into one central instance of a broker. This results in two main problems to be solved when we want to construct a digital twin out of such a distributed system. We need to build a federation of these brokers and we need to transform data into one common data model.

Given that a NGSI-LD Broker is also an NGSI-LD Context Source and that Scorpio is using the NGSI-LD registry directly for queries and subscriptions, building a federation of brokers is a simple task with Scorpio. One instance of will be setup as the Federation Broker, which is only a virtual role and not a different operation mode for Scorpio. This Federation broker will register all of its sub-brokers in its registry. All queries going to the Federation Broker will now be forwarded to all other brokers and collected at the Federation Broker (see Figure 6).

If the network setup allows it, Scorpio also provides

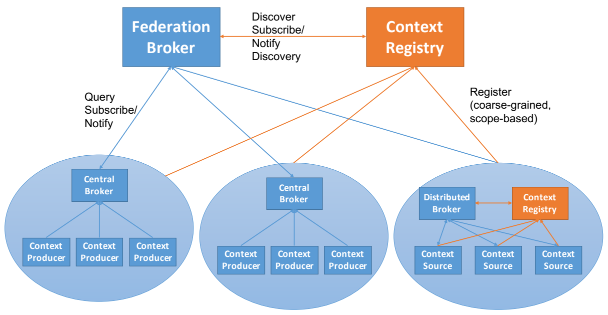


Figure 6: NGSI-LD Broker federation

a functionality for sub-brokers to automatically report new entity types to the Federation Brokers registry. While we describe a hierarchical federation here, it is also possible to build a meshed federation based on NGSI-LD. However this requires more attention to the setup of the registries to avoid circle requests. NGSI-LD and therefore Scorpio is currently not providing any safeguards to avoid circling requests. We are currently investigating possibilities to efficiently avoid such circling behavior.

NGSI-LD by itself defines only a structure for entities but no data model as such. FIWARE is a framework addressing various aspects of smart cities, smart mobility, smart agriculture etc. One of the core aspects is the context broker, which is defined as a NGSI-LD context broker [15]. What FIWARE provides on top is a collection of data models [16] describing entities for its supported scenarios. The EU has chosen the FIWARE context broker, which includes the NGSI-LD API and its data models, as its CEF context broker. Therefore it is sensible to use FIWARE data models to implement digital twins, especially in a European context. While the smart data models can be used without when setting up new data sources, there is often a need to receive or publish data in other data models. This is supported from the NGSI-LD side by using the @context entry in a document. @context is a part of the JSON-LD specification, on which NGSI-LD builds on, and is an entry storing mappings between short attribute names and full URIs as identifier for the attribute.

4.2 Edge Computing with FogFlow

FogFlow is an open source fog computing framework that can dynamically orchestrate IoT services over cloud and edges on-demand, in order to fulfill high-level *service intention* expressed by service consumers, which could be external applications or any IoT devices. Figure 7 shows a high level view of the FogFlow system. It consists of a number of *fog nodes*, each of which runs a *Broker* and a *Worker*. A management node runs two centralized components, namely *Discovery* and *Orchestrator*. Each node is a Virtual Machine (VM) or physical host deployed either in the cloud or at edges. All fog nodes form a hierarchical overlay based on their configured GeoHash IDs. All data in the system is represented as entities saved by a Broker and indexed by the centralized Discovery for discovery purposes. The data can be raw data published by IoT devices, intermediate results generated by some running data-processing tasks, or data available at a resource, reported by fog nodes. When a fog function is registered, Orchestrator will subscribe to the input data of the fog function to Discovery. Once the subscribed data pieces appear or disappear in the system, Orchestrator will be informed, and it can then take orchestration actions accordingly, which will be carried out by an assigned worker. The initial version of FogFlow provides only the centralized implementation of Discovery and Service Orchestrator. However, this centralized approach has limited scalability and reliability and also leads to a long delay of launching a new data service on the fly. This is because, with the centralized approach, all orchestration decisions must be made by the centralized service orchestrator, which is usually deployed on the cloud node of the FogFlow system. To avoid this bottleneck and also improve the scalability and reliability of the service orchestrator in FogFlow, the latest FogFlow introduces a new decentralized orchestration mechanism based on distributed discovery and orchestrator. The key idea behind the decentralized orchestration is to leverage the geoscope information associated with intents to break down the entire orchestration workload into different regions, allowing each orchestrator that runs at each edge node or in the cloud to make its own orchestrations locally.

With the decentralized service orchestration, the entire FogFlow system consists of a set of autonomous agents

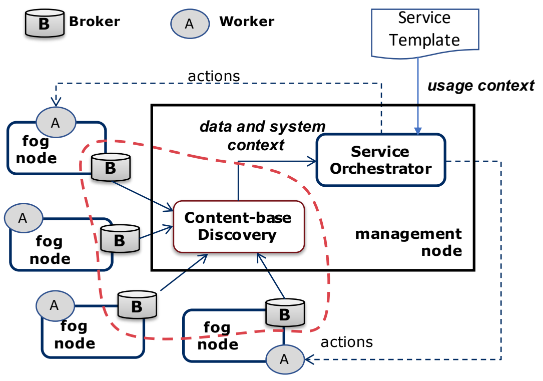


Figure 7: System Overview of FogFlow

and each of them includes the same set of components, worker, broker, discovery and orchestrator, but they are organized at different logical layers to cover different regions of the entire geomap. A global routing table that indexes which agent is responsible for which region is propagated and maintained via a gossip protocol among all agents. Every intent to trigger the service deployment will be associated with a pre-defined geoscope, which indicates which regions are covered or overlapped with this intent and then can be used to look up which agents should be involved to deal with this intent. This is all done via the distributed discovery and broker network across agents. In the end, this intent will be forwarded to all involved orchestrators and each of them will start to monitor the context information in its own local region and then make the proper orchestration decisions locally and immediately to deploy new tasks whenever the required input data becomes available from its own broker.

FogFlow is used as an advanced orchestrator to dynamically instantiate and manage all atomic services around digital twins over cloud and edges in a seamless manner. In the latest version of FogFlow, NGSI-LD is now used as the internal data model and communication protocol to exchange information between different digital twins. Also, it is able to orchestrate multiple vThings both in the cloud and at the edges, making the digital twins live closer to their physical counterparts in the root silos. More importantly, FogFlow allows digital twins to interact with each other directly at the edge when the corresponding things are close to each other. Overall, it offers three main technical benefits:

1. Reduce the internal bandwidth consumption and communication latency between digital twins and their physical things counterparts;
2. Enable *direct* communication and interactions between digital twins, seamlessly over the cloud and edges;
3. Provide the flexibility and *programmability* to realize different digital twins with its intent-based edge programming model.

4.3 Knowledge Infusion

Knowledge Infusion [18], a form of programmatic labeling such as proposed in Snorkel [23], aims to dynamically infuse weak and strong knowledge, i.e., logic based on human reasoning and internal and external knowledge bases (e.g., stored in a knowledge graph) into supervised learning to improve overall robustness, transferability and accuracy. The infusion of weak and strong knowledge has two benefits: (1) it reduces the effort needed to train or startup a ML model and (2) the knowledge model can be executed side-by-side with the ML model to correct wrong outputs, thereby improving robustness, and enabling the calculation of an uncertainty value that gives an indication of when reality and the ML model have shifted too much apart so that performance would suffer.

We have implemented KISS (Knowledge Infusion made Simple Suite), an easy tool for domain experts and knowledge engineers to infuse their knowledge in form of heuristic rules, while also utilizing existing knowledge contained in knowledge bases. Figure 8 depicts the interplay of KISS with Scorpio Broker and FogFlow cloud-edge executor. Domain experts use KISS to develop their machine learning models in an interactive Jupyter Notebook environment, on data that is ingested via an adapter to a Scorpio broker instance. After development, the trained model is packaged together with the used supervision signals (i.e., weak and strong functions, optionally a small labeled dataset) and deployed in a FogFlow pipeline [**C2**].

4.4 Schema Matching

Our data linkage platform enables schema matching between data stored in different Scorpio broker silos (see Figure 9). For the matching, we use existing linkage knowledge, for example in form of distance heuristics and based on public knowledge bases. Utilizing Knowledge Infusion we are able to jointly apply these sources to train a machine learning model that then matches the concepts from different ontologies [**C1**].

4.5 Federated Learning across Silos

Federated learning has been proposed as a promising approach to coordinating model AI training over distributed data sets without sharing original raw data, however, this approach focuses on the model training phase, rather than the data labeling phase. It has the following limitations: 1) it requires a centralized parameter server to do the fine-grained coordinating of the entire training process over all clients (there is a client running for each domain or site), but the centralized parameter server could be the bottleneck and a single point of failure for the training processing; 2) labeled data must be available on each client, which is not the case in many real world scenarios; 3) it is not model agnostic because it required the trained model to be the same kind for every client, which limits the flexibility for each domain to use and select a suitable trained model for its own domain.

Instead of moving data to labeling for learning a single global model, we propose to move labeling functions to data for learning any local model, which can still benefit

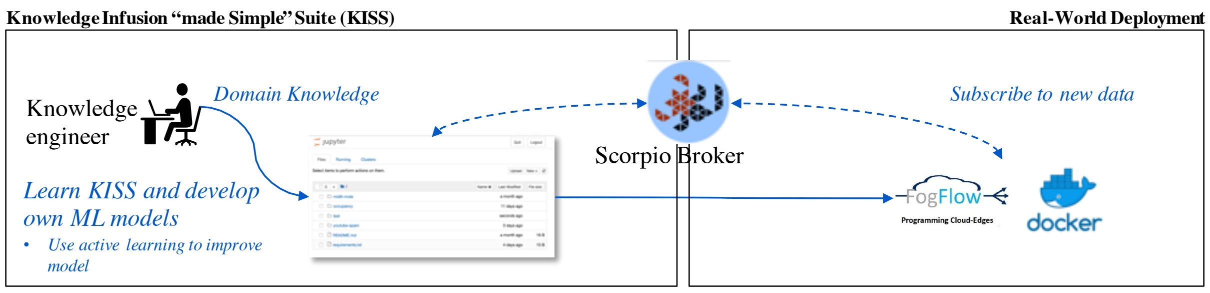


Figure 8: From ML Development to ML Execution. Domain experts can develop their ML models by utilizing existing baselines and weak supervision sources. They further provide label input via active learning. The development environment is connected with the actual deployment via Scorpio broker. FogFlow is used as execution engine for the dockerized ML models.

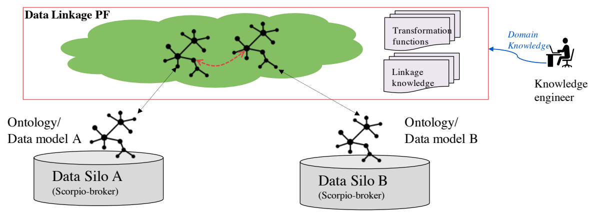


Figure 9: Schema matching. Data is being matched on the schema level across data silos using various matching heuristics and domain expert input.

from the knowledge transferred from the other domains in the label generation phase. Since the knowledge is transferred from one domain to another domain in the label generation phase by sharing labeling functions and their learned weights and estimated performance metrics, training the local model is model-agnostic and can be done with largely reduced labeling cost. In order to train a machine learning model out of unlabeled data jointly across different domains without violating privacy regulations, we introduce a federated data programming method. Based on our approach, a set of pre-defined or pre-trained labeling functions can be exchanged across domains and then each domain can dynamically select a customized set of labeling functions according to its own requirement and its local data set and then ensemble them to train its own generative model for producing labeled data, which could be later utilized to train any machine learning models. Different from traditional federated learning methods, our method does not require a centralized server to coordinate the learning process across domains and also does not require each domain to have labeled data. Also, as compared to the existing data programming approach like Snorkel [24], our method does not need to collect all unlabeled data for local training while still being able to leverage the knowledge from the other remote domains to improve the training process in the local domain by exchanging the evaluated weights and performance metrics of all label functions across domains.

Our approach has the following technical features: 1) privacy-preserving, because only the label functions and their weight are exchanged across domains but the original data stay within its own domain; 2) high efficiency, because sharing labeling functions and their evaluated weights across domains allows each domain to leverage the knowledge coming from other remote domains; 3) high scalability, because the communication cost for exchanging label functions and their weights is low and also there is no need of centralized coordinator; 4) avoid cold-start problem of traditional machine learning, because labeled data are not required and any existing knowledge can be directly used as labeling functions; 5) enable model-agnostic learning for each personalized domain and allow each domain to train its own personalized model adaptive to its own data distribution and environment.

5 Digital Twins in Action

With the technologies presented, we realize part of the use cases that has been introduced in Section 2.1. Figure 10 shows a concrete example with three digital twins related to smart parking. In such a scenario, when a connected car enters a parking house with a connected gate controller and a connected camera, two digital twins are created, one for the gate controller with the MQTT interface connected to a MQTT broker running at the edge and one for the camera with oneM2M interface connected to a oneM2M gateway running at the edge.

We orchestrate two atomic services for the twin entity of a camera: 1) a synchronization service that fetches oneM2M data and converts it to update the NGSI-LD based data presentation of the virtual camera; 2) an internal service called "Car-plate detection" that takes

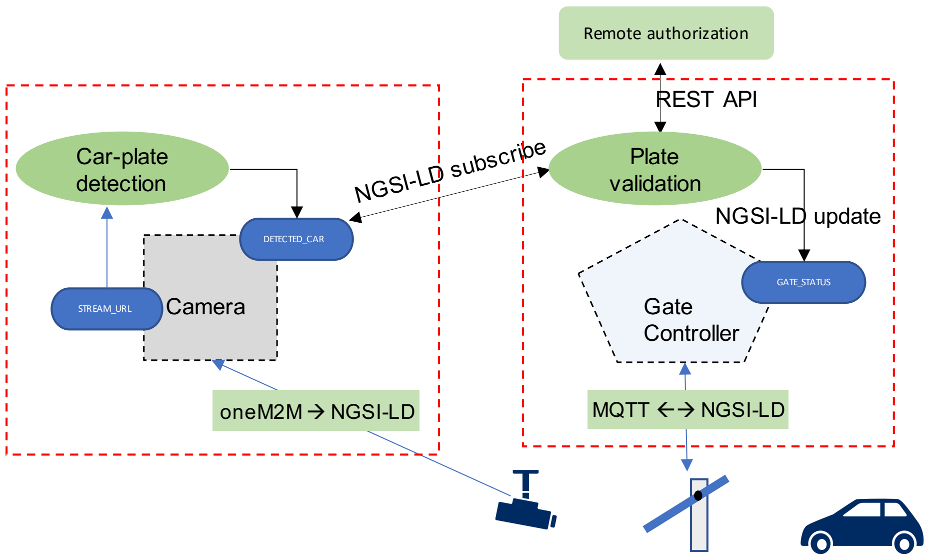


Figure 10: Two digital twins with their atomic services realized for smart parking using FogFlow and Scorpio

the "STREAM_URL" of the camera as input and then constantly reads its video stream to perform real-time car plate detection and updates the "DETECTED_CAR" property with the detected car plate number.

For the twin entity of the gate controller two other atomic services exist: 1) a synchronization service that can fetch the MQTT message reported by the gate controller and use the converted information to update the NGSI-LD based data presentation of the virtual gate controller; in the meantime the synchronization service will also subscribe the updates of the "GATE_STATUS" property of the gate controller and then write them back to the gate controller as command messages; 2) an external service called "plate validation" that can subscribe to the detected car plate number from the virtual camera and then update the "GATE_STATUS" property of the virtual gate controller after a plate validation procedure. This example shows that, with the help of FogFlow and Scorpio, a complex control process between two physical things can be accomplished fast and automatically via the communication/interaction between their synchronized twin entities.

6 Related Work

In context of urban planning the concept of digital twins has been applied in several cities, especially for providing a common 3D model of the built infrastructure, e.g., in Singapore (Virtual Singapore [20]), in Zürich [26], Helsinki [25] and Herrenberg, Germany [7, 8].

On the platform level, Conde et al. [6] propose a reference architecture, using components from the FIWARE ecosystems and its published open data models to create digital twins for any domain. Shao et al. [27] raise the need for a standardized digital twin framework for smart manufacturing. Francisco et al. [17] create a energy digital twin for cities based on data from available smart meters to benchmark energy consumption for different time periods.

From Industry, Microsoft Azure Digital Twins [19] provide a virtual integration layer for assets from disintegrated siloed data sources, while connecting to computation infrastructure and services on the Azure platform. Bosch IoT Things [2] allows applications to store and update data, properties, and relationships of physical assets based on Eclipse Ditto [13], focusing mostly on the data representation of digital twins.

Compared to these works, we propose a data driven, systematic approach to enable digital twins for cities based on a set of open and standardized components. We focus on how to enable digital twins across data silos and enable twin construction as well as internal and external services utilizing user-created ML models.

7 Outlook and Conclusion

We have presented requirements and challenges to transfer the digital twin concept to smart cities in which data silos across different organizations with heterogeneous data models are the major obstacle. We proposed to address the issue by automating the process for creating digital twins in each silo at the edge, and federating them across silos using a combination of schema matching and federation context brokers. The main components of our architecture are part of a larger open-source ecosystem which promotes open smart city standards. In the future we hope that both, open software components and open standards will enable a true digital twin transition for smart cities.

Acknowledgements

Co-financed by the Connecting Europe Facility of the European Union

This work is supported in part by the CEF ODALA project and has received funding from the European Union's Connecting Europe Facility programme under the grant agreement n^{o} INEA/CEF/ICT/A2019/2063604.

References

[1] Y. Bazilevs, X. Deng, A. Korobenko, F. Lanza di Scalea, M. Todd, and S. Taylor, "Isogeometric fatigue damage prediction in large-scale composite structures driven by dynamic sensor data," *Journal of Applied Mechanics*, vol. 82, no. 9, 2015.

[2] Bosch, "Managed inventory of digital twins for IoT device assets," https://developer.bosch-iot-suite.com/service/things/, 2021.

[3] K. Bruynseels, F. Santoni de Sio, and J. van den Hoven, "Digital twins in health care: ethical implications of an emerging engineering paradigm," *Frontiers in genetics*, vol. 9, p. 31, 2018.

[4] B. Cheng, J. Fuerst, G. Solmaz, and T. Sanada, "Fog function: Serverless fog computing for data intensive iot services," in *IEEE International Conference on Services Computing (SCC)*. IEEE, 2019, pp. 28–35.

[5] B. Cheng, G. Solmaz, F. Cirillo, E. Kovacs, K. Terasawa, and A. Kitazawa, "Fogflow: Easy programming of iot services over cloud and edges for smart cities," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 696–707, 2017.

[6] J. Conde, A. Munoz-Arcentales, A. Alonso, S. Lopez-Pernas, and J. Salvachua, "Modeling digital twin data and architecture: A building guide with fiware as enabling technology," *IEEE Internet Computing*, 2021.

[7] F. Dembski, U. Wössner, M. Letzgus, and M. Ruddat, "Urban digital twins for smart cities and citizens: the case study of herrenberg, germany," *Sustainability*, vol. 12, no. 6, p. 2307, 2020.

[8] F. Dembski, U. Wössner, and C. Yamu, "Digital twin," in *Virtual Reality and Space Syntax: Civic Engagement and Decision Support for Smart, Sustainable Cities: Proceedings of the 12th International Space Syntax Conference, Beijing, China*, 2019, pp. 8–13.

[9] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, and S. Sun, "Knowledge vault: A web-scale approach to probabilistic knowledge fusion," in *20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 601–610.

[10] X. L. Dong, "Challenges and innovations in building a product knowledge graph," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2869–2869.

[11] X. L. Dong and T. Rekatsinas, "Data integration and machine learning: A natural synergy," in *2018 international conference on management of data*, 2018, pp. 1645–1650.

[12] X. L. Dong and D. Srivastava, "Big data integration," in *29th international conference on data engineering*. IEEE, 2013, pp. 1245–1248.

[13] Eclipse Foundation, "Eclipse Ditto," https://www.eclipse.org/ditto/, 2021.

[14] ETSI, "Context Information Management (CIM); NGSI-LD API," https://www.etsi.org/deliver/etsi_gs/CIM/001_099/009/01.04.01_60/gs_cim009v010401p.pdf, 2021.

[15] FIWARE Foundation, "FIWARE: The Open Source Platform for Our Smart Digital Future," https://www.fiware.org, 2021.

[16] FIWARE Foundation, "Smart Data Models," https://github.com/smart-data-models, 2021.

[17] A. Francisco, N. Mohammadi, and J. E. Taylor, "Smart city digital twin–enabled energy management: Toward real-time urban building energy benchmarking," *Journal of Management in Engineering*, vol. 36, no. 2, p. 04019045, 2020.

[18] J. Fürst, M. F. Argerich, B. Cheng, and E. Kovacs, "Towards knowledge infusion for robust and transferable machine learning in iot," *Open Journal of Internet Of Things (OJIOT)*, vol. 6, no. 1, pp. 24–34, 2020.

[19] Microsoft, "Azure Digital Twins," https://azure.microsoft.com/en-us/services/digital-twins, 2021.

[20] National Research Foundation, "Virtual Singapore," https://www.nrf.gov.sg/programmes/virtual-singapore, 2019.

[21] NEC, "Scorpio NGSI-LD Broker," https://github.com/scorpiobroker/scorpiobroker, 2021.

[22] E. Negri, L. Fumagalli, and M. Macchi, "A review of the roles of digital twin in cps-based production systems," *Procedia Manufacturing*, vol. 11, pp. 939–948, 2017.

[23] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré, "Snorkel: Rapid training data creation with weak supervision," vol. 11, no. 3, Nov. 2017, pp. 269–282.

[24] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré, "Snorkel: Rapid training data creation with weak supervision," *The VLDB Journal*, vol. 29, no. 2, pp. 709–730, 2020.

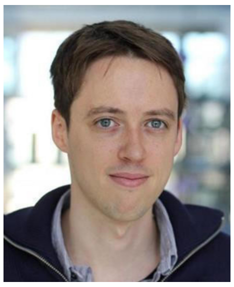
[25] T. Ruohomäki, E. Airaksinen, P. Huuska, O. Kesäniemi, M. Martikka, and J. Suomisto, "Smart city platform enabling digital twin," in *2018 International Conference on Intelligent Systems (IS)*. IEEE, 2018, pp. 155–161.

[26] G. Schrotter and C. Hürzeler, "The digital twin of the city of zurich for urban planning," *PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, pp. 1–14, 2020.

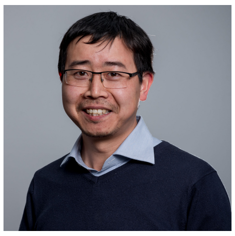
[27] G. Shao and M. Helu, "Framework for a digital twin in manufacturing: Scope and requirements," *Manufacturing Letters*, vol. 24, pp. 105–107, 2020.

[28] G. Solmaz, J. Fürst, S. Aytaç, and F.-J. Wu, "Group-in: Group inference from wireless traces of mobile devices," in *19th ACM/IEEE International Conference on Information Processing in Sensor Networks*, 2020, pp. 157–168.

Author Biographies



Jonathan Fürst is a Senior Researcher in the IoT Platform group at NEC Laboratories Europe. Previously he was a PostDoc at IT University of Copenhagen (ITU) in Denmark. He holds a Ph.D. and a M.Sc. from ITU advised by Prof. Philippe Bonnet and a B.Eng. in Industrial Engineering at University of Aalen. During his Ph.D., he spent six months as a visiting researcher at UC Berkeley in AMPLab (Software Defined Buildings group), His interests are in the area of IoT systems and practical applications such as smart buildings & cities, indoor localization and augmented reality. His current research focuses on developing better abstractions to program and run IoT applications easily and efficiently from edge to cloud and automated data integration and knowledge extraction using machine learning.



Dr. Bin Cheng is a senior researcher in the group of IoT Platform at NEC Laboratories Europe. He has been leading the design and implementation of an open source edge computing framework called FogFlow, which is a unique generic enabler to support serverless fog computing in the FIWARE ecosystem for smart cities and smart industry. He also led the design and implementation of a big data and analytics platform for the Santander city, which has been already in production use. His recent research interests include Edge AI, serverless fog computing, and knowledge extraction for IoT platforms. He had publications at several top-tier journal and conferences such as IEEE IoT Journal, EuroSys, NSDI, and IMC.

Benjamin Hebgen is a Software Engineer in the IoT Platform Research group at NEC Laboratories Europe (NLE) in Heidelberg, Germany. He has been leading the design and implementation of an open source context broker named Scorpio, which is an implementation of the NGSI-LD standard. Scorpio is a generic enabler to support context access in the FIWARE ecosystem for smart cities and smart industry. Before he was a main contributor to the NEC LeafEngine which is a sensor middleware used for digital signage. His current focus is on cloud and edge based IoT technologies for Digital Twins.