Towards a Digital Twin Framework for Connected Vehicles

Jan Gerhards, Tim Schneider and Pascal Hirmer
Institute for Parallel and Distributed Systems, University of Stuttgart, 70567 Stuttgart, Germany

Keywords: Connected Vehicles, Digital Twin, Internet of Vehicles, Autonomous Driving.

Abstract: In the last decade, vehicles have become more and more sophisticated in terms of automated driving assistance systems, safety systems, such as sleep detection, or infotainment. These systems are enabled through a growing amount of sensors and actuators built into modern vehicles, including cameras, GPS, distance sensors, or collision detection sensors, each controlled by a growing number of ECUs. Using the data generated by these vehicles, new applications can be developed, the most promising being autonomous driving. However, besides the sensing capabilities, e.g., to detect other vehicles or pedestrians, it is also necessary to provide a high interconnection of multiple vehicles in traffic. By doing so, vehicles can notify other vehicles of hazardous driving conditions, accidents, or traffic jams, leading to an Internet of Vehicles. Yet, ensuring a reliable and uniform communication and coordination of multiple heterogeneous vehicles from different manufacturers is a challenging task. In this paper, we introduce a first approach for a digital twin framework intended for connected vehicles. It enhances connected vehicles with an administration shell, making it possible for them to be recognized by other vehicles and to communicate with them.

1 INTRODUCTION

In recent years, modern vehicles have become more and more intelligent due to a growing amount of Electronic Control Units (ECU) as well as built-in sensors and actuators, including internal and external cameras, pedestrian detection sensors, distance sensors, sleep detection sensors, and so on (Coppola and Morisio, 2016). These high sensing and acting capabilities form the foundation for new applications, such as autonomous or semi-autonomous vehicles (Gerla et al., 2014; Mariani and Zambonelli, 2020), increasing the safety and comfort of passengers, pedestrians, and other traffic participants (e.g., bicycles or motorcycles). However, in order to fully utilize the capabilities of a single vehicle, it is also vital to enable communication and interconnection with other vehicles, pedestrians, and road infrastructure, such as traffic lights (Coppola and Morisio, 2016). In addition, Road-side Units (RSU) can serve as communication hubs, offer data pre-processing capabilities, or provide information to vehicles, e.g., weather or traffic information (cf. Figure 1). However, enabling modern applications in the Internet of Vehicles (IoV) (Coppola and Morisio, 2016), such as autonomous driving, is a difficult task, besides for legal and psychological reasons, also due to the high heterogeneity regarding software and hardware components, or communication protocols. Furthermore, *legacy* vehicles, providing no interconnection or sophisticated sensing and acting capabilities, further increase the already high complexity.

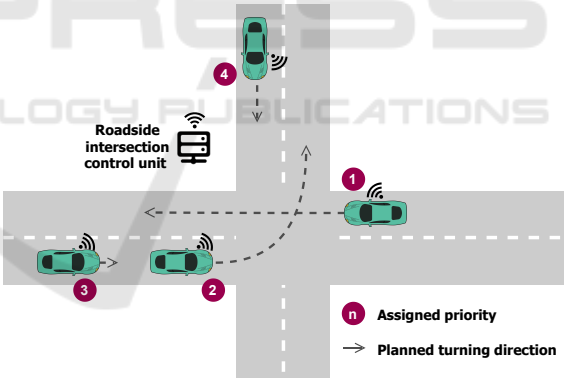

Figure 1: Example of a cooperative intersection management system.

In order to build modern IoV applications, it is necessary to create a digital representation of the physical world, including all involved connected and non-connected vehicles, pedestrians, and infrastructure components.

This digital representation is called the *digital twin* (Alam and El Saddik, 2017) in the scope of this

Gerhards, J., Schneider, T. and Hirmer, P.
Towards a Digital Twin Framework for Connected Vehicles.
DOI: 10.5220/0011066600003194
In *Proceedings of the 7th International Conference on Internet of Things, Big Data and Security (IoTBDS 2022)*, pages 189-196
ISBN: 978-989-758-564-7; ISSN: 2184-4976

paper. We define the digital twin as *a geographically independent software representation of a physical object*. The digital twin is an essential concept to realize new applications for connected vehicles, such as autonomous driving.

In this paper, we introduce a framework for a digital twin of connected vehicles. It equips different connected vehicles or other traffic participants with an administration shell, making it possible for them to be recognized and considered by each other, and furthermore, to communicate with others to share information or warn about dangerous situations. By doing so, the high heterogeneity of the different involved entities can be abstracted and an intercommunication can be facilitated. We validate our approach by a exemplary scenario, a list of requirements, as well as a prototypical implementation serving as proof-of-concept. The main contributions of this paper are: (i) a list of requirements, derived from an exemplary scenario, (ii) a generic digital twin framework, and (iii) a proof-of-concept implementation of our exemplary scenario for validation purposes.

The remainder of this paper is structured as follows: In Section 2, we first introduce an exemplary scenario, which serves as foundation for our approach. Section 3 then describes related work. In Section 4, our main contribution is described – a digital twin framework for connected vehicles. Section 5 then describes our prototype and Section 6 summarizes our paper and gives an outlook on future work.

2 EXEMPLARY SCENARIO

In this section, we introduce a exemplary scenario, which serves as foundation for the concepts of our paper. The goal is increasing time efficiency of traffic at intersections using IoV technologies. The scenario is depicted in Figure 1. Four cars are about to pass through an intersection with overlapping planned turning directions. In a world without connected vehicles, traffic light systems or static rules would now have to be applied. In IoV, on the other hand, an RSU may regulate traffic by giving priority to vehicles entering the intersection. This way, it is possible to use the exchanged sensor data to regulate traffic optimally. Beside priorities, this can be achieved by providing vehicles with concrete instructions for actions, such as their ideal speed. The RSU can also be used as a connection point to a cloud or fog computing component that, for example, may have an overall view of the total traffic situation of a city.

An optimization applied in the scenario could be that the car coming from the right gets priority over the car coming from the top due to being closer to the intersection.

The car coming from the left can already prepare the turn, while the prioritized car is crossing the intersection. Therefore, the car coming from the top gets the lowest priority and is instructed to reduce its speed so that it only arrives at the intersection when all other cars have already passed.

Based on this exemplary scenario, we derived several nonfunctional requirements that need to be considered by digital twin framework. These requirements are: (i) high scalability, (ii) coping with heterogeneity, (iii) high extensibility, and (iv) robustness.

First, since the scenario's environment is very dynamic in the sense of constantly entering and leaving vehicles, scalability is an important issue. The digital twin framework needs to be able to cope with a strong increase in vehicles, pedestrians, and other entities without causing latency issues or bottlenecks.

Second, it is of vital importance to cope with a high level of heterogeneity. Each vehicle manufacturer may use different technologies and standards. Furthermore, besides the intercommunication of vehicles, communication with other entities may also be required. In our scenario, this would be the RSU controlling priorities or smart phones of pedestrians planning to cross the intersection.

Third, the framework needs to be extensible to integrate new hardware and software technologies or network protocols. This ensures that all involved scenario entities are always able to communicate with each other.

Finally, we aim at ensuring a high degree of robustness, since IoV environments are considered safety critical. Hence, failing or leaving entities need to be discovered and their surroundings need to be informed in a timely manner. In our scenario, the failure of an entity's communication component must be communicated to the RSU aiming to give this vehicle the right of way preemptively or, in case of doubt, to indicate that conservative traffic rules should be applied. We are aware that security and privacy are also important requirements for such scenarios. However, due to the limited scope of this paper, we do not focus on them.

The aforementioned requirements aim for a solution in the context of IoV scenarios. Since many different actors have to communicate with each other in IoV, these requirements are intended to be rather generic. Therefore, the solution proposed in the paper can also be applied to other domains that have similar requirements.

3 RELATED WORK

A similar concept to our approach is the Asset Administration Shell (AAS) (I4.0, 2021), which is mostly applied in Industrie 4.0. The AAS was introduced in 2015 as part of the Reference Architecture Model for Industrie 4.0 (Boss et al., 2020). Since its introduction, AAS was often subject of further discussion and its specification was improved multiple times.

According to Dorst et al. (Dorst et al., 2016), Industrie 4.0 components can be based on objects with "*at least passive communication ability*". The intention of the AAS is to "*turn[...] an object into an Industrie 4.0 component*". In order to achieve this, an administration shell is used to surround an object. Communication between the AAS and the object uses an internal interface that can be manufacturer-dependent. A standardized external interface is used by the AAS to manage the communication with other systems. Both interfaces in combination make the concept of AAS able to be standardized, while still being able to support different functionalities of an entity (Tantik and Anderl, 2017).

The AAS structure contains a body and a header. A manifest is a part of the header and contains important administrative information regarding communication, identification, and functionality. Additionally, it contains an index of submodels, which together with a component manager make up the body of the AAS and characterize the associated asset. The body of contains the component manager, which is an interface whose main functionality is accessing the payload of the submodels (Tantik and Anderl, 2017).

Submodels are meant to be standardized according to certain characteristics. Every single one should represent an aspect of the asset. The general idea is to include multiple submodels in an AAS, making it possible to represent different aspects of an asset. If a submodel is available within the body of an AAS, it can be assumed that some relevant properties are available (Bedenbender et al., 2017).

In summary, the AAS has been explored intensely in the context of the manufacturing industry. Within the domain of connected (and possibly autonomous) vehicles, however, only few concepts regarding framework of a digital twin are available. Instead, it is important to note ways in which vehicles may communicate in the future and what information needs may arise. Two important concepts are the Vehicular ad hoc Network (VANET) and the Vehicular Cloud, which are discussed by Gerla et al. (Gerla et al., 2014). This shows that connected vehicles may, in theory, be capable of collaborating and acting as intelligent agents.

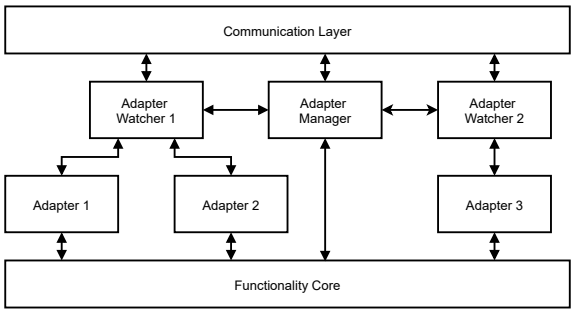

Figure 2: Example instance of a digital twin.

A different concept illustrating possibilities of this domain is proposed by Mariani and Zambonelli (Mariani and Zambonelli, 2020). In it, the authors discuss using different levels of autonomy depending on the current traffic situation. When using this concept, connected vehicles may have different information needs during operation, depending on the currently chosen level of autonomy. Therefore, a digital twin has to take into account changing information and communication needs, which are not known at the beginning of a route.

Our concept aims to consider the wide variety of approaches proposed for connected vehicles. The goal is to provide a reference framework for a digital twin designed specifically to handle relevant complexities. Elements of this framework are based on ideas related to AAS. Especially the use of different submodels for providing information relevant to separate aspects is interesting for us. Our concept combines this approach with the possibility to manage a digital twin in a constantly changing environment.

4 DIGITAL TWIN FOR CONNECTED VEHICLES

4.1 Framework

An exemplary digital twin based on our proposed framework is depicted in Figure 2. The general structure is influenced by the AAS. A digital twin should be able to support multiple interfaces at the same time. These have to be managed according to current information needs and the state of the environment. In addition, the framework includes components capable of providing functionality and data that is relevant to other traffic participants. To achieve this, our approach revolves around different *Adapters* which are managed by an *Adapter Manager* and make functionalities contained in a *Functionality Core* accessible using a well-defined interface.

4.1.1 Functionality Core

The *Functionality Core (FC)* is the part of the digital twin that contains data and functionality concerning the vehicle and its environment. Any logic required to obtain, process, and store data (like sensor handlers) is part of the FC. If decision-making is available for the vehicle (e.g., for autonomous driving), any relevant logic is encapsulated within this component. Furthermore, software used to control the vehicle itself (e.g., cruise control, high beam control) is also part of the FC.

The internal structure of the FC is mainly a black box in order to allow developers to use approaches they prefer when implementing logic used to control a vehicle. Communication partners of the digtal twin are not able to communicate with the FC directly. Instead, they use interfaces provided by *Adapters*.

If developers choose to include cloud infrastructure (e.g., for processing or communication), the FC is divided into a *Local Fragment (LF)* and any number of *Cloud Fragments (CF)*. The latter are an abstraction for cloud interfaces the vehicle hardware interacts with. In order to account for data exchange of different systems located in a cloud environment, CFs are allowed to communicate with each other. A notable use case for CFs is fleet management. In our exemplary scenario, it would be possible that some connected vehicles interact with a cloud analyzing traffic patterns. If a vehicle registers a delay at the intersection, other members of a fleet can be notified of the delay and may be able to search for a more efficient route. Such a system should be realized using a CF.

4.1.2 Adapter

An *Adapter* enables devices to communicate using a specific interface. This interface is unambiguously defined and based on messages. It characterizes how devices can interact with each other and is identified by a unique ID.

Different kinds of Adapters can be used by the digital twin. They can be capable of connecting not only to nearby connected vehicles, but also other types of devices. For example, in our scenario, an interface has to be defined for communication with RSUs. These could then be used by connected vehicles to simplify assignment of priorities at intersections. An Adapter always offers exactly one interface and transforms calls to it into instructions for the FC. This means that an Adapter bridges the black-box implementation of the FC and a known standardized interface, thereby enabling cooperation with other participants. Since all data used by Adapters is stored in the FC, the state of an Adapter consists only of information on currently available quality of service (QoS) attributes or communication channels.

When two digital twins include the same Adapter, they are able to communicate using its associated interface. One advantage of this approach is that a digital twin can provide multiple Adapters, which may be active at the same time. As a consequence, the connected vehicle will be able to exchange information and coordinate with different devices in different ways. Furthermore, it is possible to communicate with the same device using multiple Adapters. This enables the use of different interfaces at the same time. An example for such a use case may be that two cars use one Adapter for coordinating their relative speed, while simultaneously sharing information on traffic conditions using another.

4.1.3 Adapter Watcher

Adapter Watchers (AWs) are components responsible for observing the operation of Adapters. In order to achieve this, every Adapter is associated with exactly one AW. As shown in Figure 2, an AW may be connected with multiple Adapters at the same time. This may be advantageous if very similar Adapters are provided by the digital twin. One of the main tasks of AWs is to route outgoing messages of their Adapters to appropriate communication channels. This requires AWs to evaluate which channels may be used by which Adapter when communicating with which partner. Furthermore, QoS attributes may have to be taken into consideration when choosing a communication channel. While a connection is active, the AW of an Adapter may analyze messages in order to gain information about its state. This includes the outgoing messages sent by the Adapter as well as incoming messages sent by its communication partner. In addition to analyzing them, an AW may modify the content of a message, e.g., if an Adapter expects to receive only application layer data, its associated AW may remove all other layers from a message.

Another way for the AW to collect information about an Adapter is by receiving it directly, e.g., in case of an error. These notifications are then aggregated with the data gained through message analysis. Based on this, the AW is able to provide data to the *Adapter Manager* on request or by notifying it directly. In our scenario, this is how digital twins are able to detect that a vehicle is no longer communicating. If no more messages are sent, AWs pick up on this and initiate their digital twins' responses.

Finally, an AW is able to send commands to its associated Adapters. As Adapters store only little data and should be based mainly on the state of the FC,

these controls are very limited. An example of a use case for them is turning the Adapter on and off.

4.1.4 Communication Layer

The *Communication Layer (CL)* is the component of the digital twin that makes it possible to exchange messages with other devices. Any software component used to transmit or receive information is part of it. Internally, the CL is a collection of various communication channels. Each of these channels is capable of sending or receiving messages using a specific protocol and technology. For example, there may be a communication channel capable of exchanging information via 5G and another using LoRa. To identify different channels, a unique character sequence is used. When digital twins exchange information about communication channels, these identifiers are used to make intentions and data unambiguous.

When receiving a message, the CL is tasked with forwarding it to its specified destination. If no connection for the recipient exists, the Adapter Manager is informed. In order to decide which message should be passed to which component, the CL is informed about existing connections. This includes information on which communication channels have been assigned to which Adapters and partners. A situation illustrating this idea can occur when a digital twin communicates with a partner using multiple Adapters. Depending on existing connections, a communication channel may pass on messages sent by the partner and addressed to a specific Adapter. However, those specifying a different destination may not be forwarded if the CL was not informed that the channel can also be used for this other Adapter.

In the context of connected vehicles, communication has to fulfill various different requirements. Some examples for these are reliability and latency characteristics. However, it is not possible to make any guarantees based on the architecture of the framework. Instead, the developer implementing a communication channel for use in the framework has to make sure all requirements are fulfilled. When these attributes are defined, the framework is able to take them into account. Adapter Watchers should route messages to an appropriate communication channel based on QOS attributes requested for messages by the Adapters sending them. Furthermore, the Adapter Manager can consider the different guarantees of available communication channels when deciding which channels should be used to establish a connection. This way, the framework can include handling of QOS of communication channels, even though the guarantees have to be made by communication channels and their developers.

4.1.5 Adapter Manager

The component controlling Adapters and connections of the digital twin is the *Adapter Manager (AM)*. Its main task is the communication with AMs of other digital twins. Together, they decide which Adapters and communication channels to use. In order to do this, the AM is aware of the available Adapters. It is able to prioritize them based on static knowledge (e.g., their optimal use case) as well as the current state of the digital twin (e.g., number of currently ongoing connections using a specific Adapter).

During operation, the AM keeps track of connections and can react to events that might arise. For example, in our exemplary scenario, an AM may ask other AMs to connect an Adapter able to coordinate a reaction to a vehicle that stopped communicating. To command other parts of the digital twin, the AM is connected to the CL and all AWs. Adapters can be controlled indirectly via their associated AW.

Furthermore, the AM is connected to the FC. This connection is bidirectional, meaning the AM can request information from the FC and vice versa. The AM uses this connection to gain information on the state of the environment and the vehicle in order to determine the optimal configuration regarding the connections of the digital twin. This includes both the Adapters and communication channels used as well as with which digital twins connections should be established. Additionally, the FC is able to request information from the AM. For example, the amount of ongoing connections may be a relevant data point when trying to assess traffic density.

4.2 Goal of the Framework

The main goal of our framework is to isolate logic regarding the vehicle and its environment in the FC and allow access using different interfaces. An advantage of this approach is that it is possible to support different concepts using Adapters. The FC can be implemented independently of concepts used by interfaces provided to other digital twins, while Adapters include logic to transform the internal view into one compatible with their respective interfaces. In addition, different communication channels and CFs allow for different modes of communication. Furthermore, it is possible to develop Adapters and communication channels independently of one another. This facilitates separate development of different interfaces and technologies in short cycles. In the context of this domain, vehicle manufacturers could use this to their advantage by being the first to bring new features on the market or react to a change in demand.

Communication is possible on two different levels using the proposed framework. First, a connected vehicle should be able to exchange messages with other devices in its vicinity. This is possible using the CL. It contains communication channels, which support different technologies used to communicate to other devices. The use of these channels is managed by the AM, which matches channels and Adapters together based on compatibility and other factors. Second, communication with cloud infrastructure is also possible using the framework. This is done by allowing for communication with a system located in a cloud as part of the FC. Internally, its CFs are used to model a connection to an outside system related to the connected vehicle.

4.3 Digital Twin Communication

The interaction of digital twins using this framework is based on messages. This allows use of a variety of technologies and approaches for Adapters. AMs also communicate based on messages. Every digital twin receives messages using its CL, which distributes them to the appropriate component.

Furthermore, communication between Adapters of digital twins is based on messages. This is done in an effort to keep the concept simple while still allowing for all methods of remote communication to be applicable. Using, e.g. RPC, an Adapter implementing this concept would have to transform messages into the provided calls. Then, it would use FC calls to receive a result and return it using RPC messages.

The AM is responsible for managing the connections a digital twin engages in. This includes decisions, such as with which devices a connection should be established and which Adapters should be used. All coordination with other digital twins takes place using the messages defined for the AM. As a way to react to changes of the environment, the AM is able to communicate with the FC. Since the AM does not contain information on the state of the environment, this connection is used to assess the need for different connections and connection partners. Vice versa, it is possible for the FC to request data on the aspects managed by the AM, if this is needed for its operation. Apart from the FC, the AM is also in contact with all AWs and the CL. This enables it to receive information on other aspects of the digital twin, which can be considered in its decision-making.

4.3.1 Message Format

When a digital twin receives a message, it has to determine both the sender and the addressed component. This is necessary in order to check if the use of the communication channel is allowed or if the message is invalid. The first 128 bits of a message are an ID used to identify the sender. Every device has a unique ID enabling digital twins to quickly recognize each other. After the sender ID, another 64 bits are used to identify the target Adapter. For every Adapter, a unique ID is defined. This makes it possible to pass messages on to the interface they are designated for. If the target is the AM, the Adapter ID is 0.

After these two IDs, the actual message content begins. This can include any number of bits in any format, though, this may depend on the communication channel. The goal of this approach is to give extensive freedom to developers regarding the structure and content of messages they use. Consequently, many different data formats can be used for interfaces implemented by Adapters.

4.3.2 Adapter Manager Messages

Messages that can be processed by the AM are called *Adapter Manager Messages (AMM)*. An example for such a message can be seen in Listing 1. In our prototype, AMMs are encoded in JSON using UTF-8. The reason for this is that a human-readable message format makes logging and bugfixing easier. Especially in the domain of connected vehicles, this is very valuable, as identifiers can be understood across different developer teams. The information about the concrete structure of the messages is fixed in our prototype, however, it could also be derived by underlying knowledge models, e.g., using JSON-LD.

Overall, a format other than JSON could be chosen if the framework is to be used in practice. However, it is important that every digital twin instance uses the same format for AMMs.

In order to check which messages should be processed, a sequence count is used. Since no guarantees regarding the order of arrival of messages has to be made for AMMs, the AM has to account for them arriving out of order. In these cases, the sequence number in conjunction with the content of a message can be used to find out if it should still be processed.

For example, if a message arrived with a request contradicting a message with a higher sequence count, the earlier message would be ignored. Every AMM contains a field indicating its type. This is used by AMs to determine the format of the rest of the message. Depending on the type, different fields are included containing the required parameters. Furthermore, depending on the state of the connection, an AM may be able to immediately recognize invalid requests based on their type.

```
{
 "id": "472656947346",
 "sequence": "3",
 "type": "connect",
 "version": "1.2.3",
 "adapters": [
  "100",
  "7",
  "389"]
}
```

Listing 1: Example AMM.

4.3.3 Including Other Traffic Participants

In addition to vehicles, other entities/traffic participants are to be considered as well. These include (but are not limited to) pedestrians, traffic administrators, or road operators. In our exemplary scenario, the RSU is an example of such a participant. Furthermore, computing-related entities, such as (edge)cloud servers, may be relevant. These entities can also be represented using the introduced framework as long as they are able to communicate. For example, pedestrians may use their smart phones to exchange information and coordinate with their surrounding.

In addition to participants being included using their own digital twin, CFs can be used to connect stakeholders to digital twins using a static interface. Especially large systems which use a well-known interface for communication can provide a CF to exchange data with many digital twins. Using this data, it may then be possible to determine optimizations and inform traffic participants of these options.

5 PROTOTYPICAL VALIDATION

One of the goals of the prototype was to demonstrate how to incorporate cloud environments into the framework. In order to achieve this, the Multipurpose Binding and Provisioning Platform (MBP) was used to provide such an environment. This is an IoT platform introduced by Franco da Silva et al. (Franco da Silva et al., 2020). The reason for choosing the MBP is its ability to display complex JSON objects. As the AMMs are formatted in JSON, including MBP is very simple and reduces the overall complexity of the digital twin. We used JSON in our prototype, since we consider it a fitting format for exchanging messages among the involved entities. However, it is important to note that other data formats could also be used, depending on the scenario our framework is applied to, as long as the format is uniform and all involved entities agree on using it.

The digital twin itself was implemented using the programming language Python, mainly due to its native JSON support. A VM running Ubuntu was chosen as the environment in which the digital twin instances were run. In order to show how digital twins interact, multiple instances of the same code were executed. These instances then exchanged messages with each other. It is possible to configure the prototype instances to react differently to certain events. This makes it possible to show interactions between digital twins with varying behavior. The configuration and other commands are controlled using a command line interface. As a way to illustrate how CFs work, the MBP instance runs on a different system than the rest of the digital twin. In our implementation, a Windows computer was used to install and execute MBP.

5.1 Implementation of the Scenario

In our exemplary scenario, different vehicles should coordinate at an intersection. In order to achieve this, all participants have to provide Adapters which make the exchange of relevant information possible. In theory, vehicles may be able to solve this challenge using only Vehicle-to-Vehicle communication. However, in this scenario an RSU is utilized as a way to reduce the amount of connections every vehicle has to maintain. Due to this, every digital twin has to implement at least one Adapter that is also provided by the RSU. Furthermore, at least one of the communication channels offered by the RSU has to be implemented in order to exchange messages.

There are some requirements regarding the FC of participating digital twins. While the way they are implemented can be chosen freely, the necessary functionality has to be included. In our scenario, this is mainly the ability to coordinate the use of the intersection. As mentioned in Section 4.1.1, some connected vehicles may notify a cloud system of waiting times at intersections. In these cases, the FC has to include an appropriate CF.

AWs are used to oversee the state and operation of Adapters. In our scenario, this may be used to recognize vehicles that stopped communicating. If a digital twin does not send messages, AWs of its communication partners will be able to pick up on that by stopping the time they last received a message. If a certain threshold is passed, the AW may inform the AM, who can then react to the problem. For example, present vehicles may connect using an Adapter to communicate about the erroneous participant and how to behave in a safe manner while still trying to minimize waiting times.

5.2 Validation

The proposed framework supports loose coupling between components. This makes it possible to extend a digital twin without a large amount of effort. Furthermore, Adapters and communication channels as used in the framework make it possible to cope with a highly dynamic environment, such as the one connected vehicles operate in. As different interfaces can be used in different situations and a multitude of communication technologies can be supported, heterogeneity may even become an asset for developers using the concept. For example, a digital twin implementing many different Adapters may be able to more appropriately react to a given situation. Also, different communication technologies can be used based on the current requirements.

Apart from supporting connections with devices in the vicinity of a vehicle, this framework also allows for parts of a digital twin to be located in the cloud. This supports many different approaches, for example, ones which rely on large processing power of powerful machines in the cloud. Other cloud applications can also be included, for example a cloud solution which receives vehicle data and makes it accessible to the owner using a mobile app.

In addition to interoperability, a big advantage of the FC being used to encapsulate the internal logic and decision-making is the improved adaptability. As long as the interface provided for Adapters supports the required functionalities, little to no changes are expected to be necessary when adding new Adapters. Hence, a FC may be used for a long time without going out of date. Even if other technology for communication or interfaces become popular, logic in Adapters can be used to bridge old and new interfaces.

Furthermore, the FC can be upgraded independently of Adapters and communication channels. This enables developers to use new concepts in the internal logic. If older interfaces would normally not be compatible with them, Adapters can be used to transform the data as necessary. As a consequence, it is possible to improve the internal logic of the FC without affecting the interfaces provided to communication partners. This can be used, e.g., to prepare for the introduction of new features or to improve internal efficiency.

6 SUMMARY AND OUTLOOK

In this paper, we present a first approach of a digital twin framework, which can be applied to various scenarios, e.g., in the IoV. An exemplary scenario serves as a foundation for a list of requirements of the digital twin we aim for. Based on these requirements, we introduce our framework and show how communication can be enabled among heterogeneous vehicles. We validate our approach based on a prototypical implementation of our exemplary scenario.

We are aware that security and privacy are core aspects of such architectures. Due to the limited scope of this paper, this is not discussed here and is part of our future work. Another issue which could be added is the discovery of devices by the digital twin.

REFERENCES

Alam, K. M. and El Saddik, A. (2017). C2ps: A digital twin architecture reference model for the cloud-based cyber-physical systems. *IEEE access*, 5:2050–2062.

Bedenbender, H., Billmann, M., Epple, U., Hadlich, T., Hankel, M., Heidel, R., Hillermeier, O., Hoffmeister, M., Huhle, H., Jochem, M., et al. (2017). Examples of the asset administration shell for industrie 4.0 components–basic part. *ZVEI white paper*.

Boss, B., Malakuti, S., Lin, S., Usländer, T., Clauer, E., Hoffmeister, M., and Stojanovic, L. (2020). Digital twin and asset administration shell concepts and application in the industrial internet and industrie 4.0. *Plattform Industrie*, 4.

Coppola, R. and Morisio, M. (2016). Connected car: technologies, issues, future trends. *ACM Computing Surveys (CSUR)*, 49(3):1–36.

Dorst, W., Glohr, C., Han, T., Knafla, F., Loewen, U., Rosen, R., Schiemann, T., Vollmar, F., and Winterhalter, C. (2016). Implementation strategy industrie 4.0-report on the results of the industrie 4.0 platform. *Bitkom/VDMA/ZVEI*.

Franco da Silva, A. C., Hirmer, P., Schneider, J., Ulusal, S., and Frigo, M. T. (2020). MBP: Not just an iot platform. In *2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 1–3. IEEE.

Gerla, M., Lee, E.-K., Pau, G., and Lee, U. (2014). Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds. In *2014 IEEE world forum on internet of things (WF-IoT)*, pages 241–246. IEEE.

I4.0, P. (2021). Industrie 4.0 glossary. https://www.plattform-i40.de/IP/Navigation/EN/Industrie40/Glossary/glossary.html. [Online; accessed 16-September-2021].

Mariani, S. and Zambonelli, F. (2020). Degrees of autonomy in coordinating collectives of self-driving vehicles. In *International Symposium on Leveraging Applications of Formal Methods*, pages 189–204. Springer.

Tantik, E. and Anderl, R. (2017). Integrated data model and structure for the asset administration shell in industrie 4.0. *Procedia Cirp*, 60:86–91.