IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received November 20, 2020, accepted December 8, 2020, date of publication December 18, 2020, date of current version December 31, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3045856

Data Link for the Creation of Digital Twins

RIKU ALA-LAURINAHO, (Graduate Student Member, IEEE), JUUSO AUTIOSALO, (Graduate Student Member, IEEE), ANNA NIKANDER, JOEL MATTILA, AND KARI TAMMI
Department of Mechanical Engineering, Aalto University, 02150 Espoo, Finland

Corresponding author: Riku Ala-Laurinaho (riku.ala-laurinaho@aalto.fi)

This work was supported by the Business Finland under Grant 8205/31/2017 (DigiTwin) and Grant 3508/31/2019 (MACHINAIDE).

ABSTRACT A digital twin offers various features such as visualizations, simulations with real data, and performance monitoring. Several of these features or parts of them can be implemented employing existing systems storing product data. However, to create a digital twin, these data being scattered to different systems is needed to be merged. For merging data and straightforward linking of these systems, this paper proposes a Data Link. The Data Link offers a single interface to access the systems via an API (Application Programming Interface) gateway and makes all data of the physical product available and accessible. In addition, it stores metadata about the systems and offers a user interface that allows searching the system that contains the required piece of information or implements the needed feature. The Data Link was implemented for an industrial overhead crane bringing operational data, control, and CAD (computer-aided design) models behind a single interface. The example implementation indicated that the Data Link based architecture for digital twins allows easy implementation of digital twins by using existing systems.

INDEX TERMS Data link, digital twin, industrial communication, industrial internet of things, cyber-physical systems.

I. INTRODUCTION

Digital twin is a virtual counterpart of a physical entity described to reflect its state in real-time [1]. Digital twin (DT) enables predictive maintenance and optimization of operation [2] and allows designers to improve products based on operational and service data [3]. It brings all data from a physical entity available and accessible via a single interface [4]. Autiosalo *et al.* [5] identified the features of digital twin and propose a Feature-based Digital Twin Framework (FDTF) to create digital twin instances. In the proposed framework, the Data Link connects the features of a digital twin and makes the physical product information, which is currently scattered around different systems [4], available. This paper builds on the FDTF presented by Autiosalo *et al.* [5]. It further develops the Data Link concept and presents an implementation of the Data Link.

Each feature of a digital twin can be implemented by a separate system or service analogously to microservices architecture [5], [6]. Building a digital twin using microservices architecture allows better scalability and maintainability, and, robustness is improved since the failure of a single service does not likely cause the failure of the whole system [7] – in this case, the digital twin. Implementation of digital twin based on microservices is demonstrated, for example, in [8] and [9]. The major benefit of FDTF and building DT using (micro-)services is that existing systems can be used to implement features of a digital twin [5]. This makes the implementation of a digital twin faster and more cost-efficient.

The concept of Data Link was first presented by Autiosalo *et al.* at the conceptual level in [5]. However, the description of the Data Link did not include implementation details. The current paper develops the concept further by presenting a way to implement the Data Link in practice. In addition, the paper includes proof of concept implementation of the Data Link for an overhead crane. The idea of the Data Link is to provide a single access point to all data available on the physical entity (Fig. 1). This paper proposes an API (Application Programming Interface) Gateway for linking the systems (or features) that form a digital twin. The API gateway simplifies the communication between these services by forwarding messages and taking care of authentication. Data Link provides information about

The associate editor coordinating the review of this manuscript and approving it for publication was Yassine Maleh.

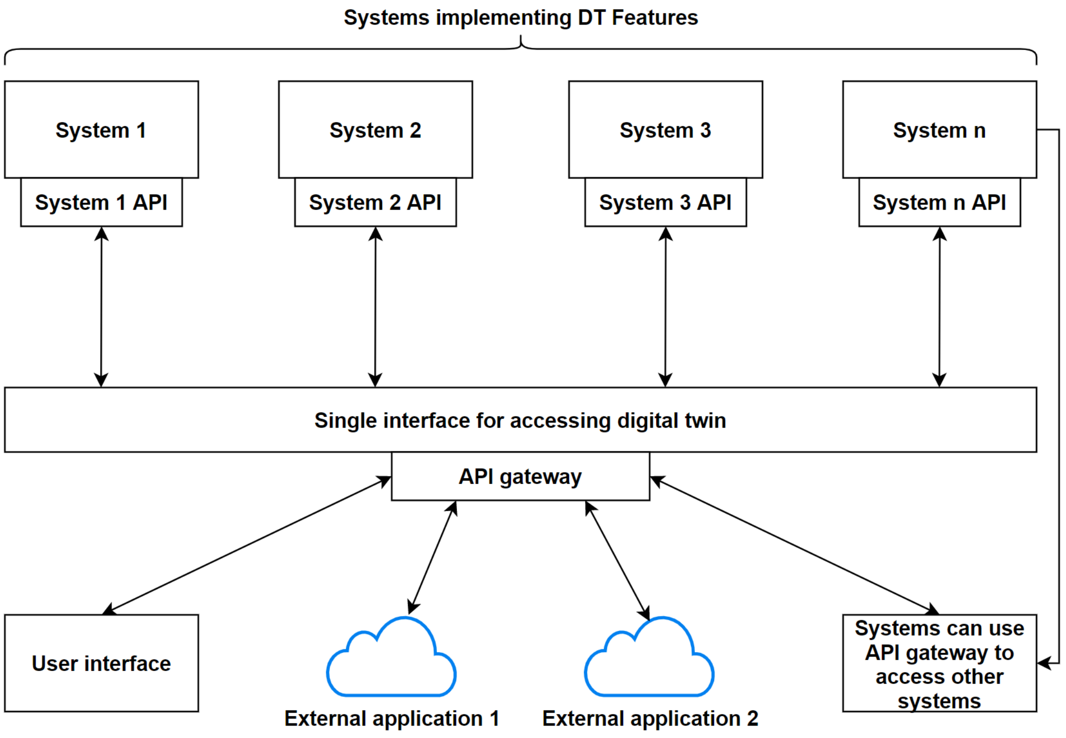


FIGURE 1. Data Link connects systems that implement the features of the digital twin behind a single interface.

the digital twin and its features via a user interface. The information is stored in a YAML document, the format of which authors aim to standardize to allow the implementation of Digital Twin Web (DTW). DTW is analogous to the document-based World Wide Web (WWW) but made for digital twins. This paper contributes to DTW development by building an initial version of an overhead crane "DT document".

In this paper, an overhead crane called Ilmatar [10] is used as an example of an industrial machine for which a digital twin prototype is created following FDTF. Currently, the information of the crane is scattered across several systems that can be used to implement DT features. For example, Siemens MindSphere and Teamcenter store historical data and product design data such as CAD models, respectively, OPC UA server allows access to real-time data and control, and TRUCONNECT contains crane condition and operational data.

The main contributions of the paper are as follows:

1) Develop further Data Link concept presented in the Feature-based Digital Twin Framework by Autiosalo *et al.* [5]
2) Build a digital twin prototype for an overhead crane following FDTF and using Data Link
3) Introduce a DT document standard draft to describe the features of a digital twin

The first contribution is discussed in sections IV, VI and VII-A, the second contribution in V and VII-B, and the third in III and VII-C.

II. RELATED WORK

The origins of the digital twin concept are on the PLM field, and it was first introduced by Grieves in 2002 [11]. Another early adopter of the digital twin concept was the aerospace field, and NASA planned to use the digital twin to simulate and predict the vehicle behavior [12]. In both early visions, the digital twin mirrors the physical counterpart accurately, ultra-realistically [12] or "from the micro atomic level to the macro geometrical level" [11]. After that, the concept has significantly developed, and a high-fidelity model of the physical product is not anymore a mandatory part of a digital twin. For example, a definition by Industrial Internet Consortium: "A digital twin is a formal digital representation of some asset, process or system that captures attributes and behaviors of that entity suitable for communication, storage, interpretation or processing within a certain context" [13], allows almost any digital representation linked to an entity to be called a digital twin. This paper continues the work presented in [5] and share its definition of a digital twin: "Digital twin is a virtual entity that is linked to a real-world entity. Digital twin consists of various features that are selected and customized to serve the needs of diverse use cases." Several review papers have been recently published about digital twins [14]–[17] that provide more a comprehensive view on the origins and characteristics of the digital twin.

The Feature-based digital twin framework was developed by Autiosalo *et al.* [5] as a way to structure digital twins and to put a digital twin from an abstract definition into practice. The idea of the framework is to construct digital twins from separate software blocks, each providing specific functionality for the digital twin. The software blocks are connected with a Data Link that offers a single interface for accessing the digital twin components. This approach allows creating more complex digital twins because the twin is not restricted to the capabilities of any single software platform. As a limitation, the study did not provide implementation details for the Data Link.

Employing microservices in the building of a digital twin has been proposed by several papers [6], [8], [9], [18]. Alasaam *et al.* [8] used Apache Kafka to enable stream data processing with microservices. Kafka also offered an API to, for example, read and write stream data. This approach is suitable for handling a continuous stream of measurement data from sensors attached to a physical product. However, it is not optimal for connecting various features of a digital twin, because for each feature an adaptor for Kafka has to implemented. Borodulin *et al.* [6] introduced the Digital Twin Cloud Platform concept, in which a digital twin is described as a set of services which implement the features of a digital twin such as data analysis and simulation. These features could be accessed via API providing them as microservices, and the idea of the concept is similar to the FDTF. However, the communication model is not described in detail, and sufficient information for the implementation of the concept is not provided. Preuveneers *et al.* [9] introduced a digital twin architecture based on microservices, which allows easy addition of new features to the digital twin. To prevent local failures from propagating, the Digital Twins can be toggled by other DTs via RESTful interface. The paper does not describe in detail how the microservices communicate and if it is possible to use already existing systems for creating a digital twin. Mena *et al.* [18] presented the Digital Dice concept that relies on microservices and is based on a digital twin. Microservices are responsible for implementing the features of the Digital Dice, and a Digital Dice can be generated from the Web of Things (WoT) Thing Description (TD) document. The paper does not consider the applicability of the Digital Dice architecture for more complex digital twins with more advanced features. In addition, the implementation of the communication with microservices is not described in detail. Thramboulidis *et al.* [19] proposed a framework for the use of microservices architecture in cyber-physical manufacturing systems. The microservices communicate using the LwM2M (Lightweight Machine to Machine) protocol that runs on top of CoAP (Constrained Application Protocol) and their properties are described in CoRE resource directory. Compared to FDTF, each physical unit of the manufacturing plant has only one corresponding microservice. We argue that with complex physical products it is necessary to use several microservices to build a digital twin to avoid creating of one "monolithic" microservice.

Haag and Anderl presented a proof of concept digital twin for a bending beam [20]. The communication between the features of the digital twin followed the publish-subscribe model and was implemented with the MQTT protocol. Publish-subscribe communication is suitable for use cases in which information from several data sources (publishers) is needed to be distributed to several clients (subscribers). In our case, data is often requested by one service from another service, and, therefore, the publish-subscribe communication model is not beneficial.

In FDTF, the services which implement the features of digital twin should have well-defined APIs. On the other hand, the digital twin should also have an API that offers an interface to access these services. Scheibmeir and Malaiya [21] presented guidelines for developing DT software, including APIs, such as Test-Driven Development. In addition, they proposed an API mediation layer for enabling security, management, and metering. The functionalities of the API mediation layer can be implemented with the Data Link in FDTF. To enable interoperability of digital twins, Platenius-Mohr *et al.* [22] proposed file- and API-based information exchange in which source information is translated to target information using intermediate mapping model. In the proposed approach, the data of a digital twin can be accessed using REST API. This approach could also be used on top of FDTF to promote interoperability. Ciavotta *et al.* [23] presented a microservice-based middleware that offers a Web User Interface and API gateway for managing digital twins in smart factories. In this model, digital twins are not composed of microservices, but the platform they are part of uses microservices to implement its functionalities.

For describing a digital twin and its features, a few approaches have been proposed. Mena *et al.* [18] used Web of Things Thing Description document to depict the IoT device features and endpoints to access these features. In addition, a method to create a Digital Dice (or a digital twin) from the TD document was introduced. Web of Things architecture developed by World Wide Web Consortium (W3C) aims to enable interoperability of IoT devices [24]. Thing Description document includes device metadata, interfaces, and relations to other Things expressed with web links [25]. This information is described in a human-readable and machine-understandable way using JSON and JSON-LD. The vocabulary used to define terms in the TD document can be extended with TD Content Extensions [25] which allows the creation of a digital twin specific vocabulary.

Microsoft has published Digital Twin Definition Language (DTDL) [26]. They use it for Azure digital twins and provide it as an open standard that enables communication with other software providers' products. We see DTDL as a promising development direction. However, the identifiers it provides are not globally unique by default, which, in our opinion, tends to create siloed digital twin networks. Also, even though the identifier documentation has a URI compatibility section, it, unfortunately, leaves the reader unaware of how to form a URI based on the identifier. Nevertheless, we see DTDL as the most promising of existing initiatives for creating a global network of digital twins and will be following it closely.

III. DIGITAL TWIN WEB (DTW)

Digital Twin Web (DTW) is an ongoing development effort for creating a global network of digital twins in a similar internet-native and user-friendly manner as the World Wide Web (WWW) provides information to people. DTW is planned to consist of any means necessary to achieve this goal, for example, a language or a schema for defining the core properties of digital twins and a protocol for twins to communicate with each other. DTW development was initiated by the FDTF framework [5], and especially the descriptions of the "Data Link" and "Identifier" include basic design principles for DTW. The current paper leverages the DTW as a long-term end goal, providing motivation for incremental concrete actions.

We aim the DTW to use and integrate with as many existing standards as possible, e.g., using the WWW schemas (http:// and https://) as default and creating a specialized digital twin schema only if the need for that may be traced to fundamental basic principles, i.e., it would not be possible to build a global network of digital twins without it. We acknowledge that human adoption represents an important role in these fundamentals. Hence, even though some design choices may seem disadvantageous from a technical perspective, they may be justified to speed up the human adoption rate of DTW. We see poor usability as a major issue in IoT standards: using them requires too much learning, leading to a situation where only a limited amount of people have the time available to learn them, meaning that the standard will not reach wide adoption.

In DTW, each digital twin is described with a "DT document" containing the identity, basic information, and the connected services of the digital twin. The need for this type of document was also identified when the Data Link was developed and the connected features were needed to be described. The authors aim to initiate a standardization process for this document. However, the result of the standardization does not necessarily have to be a completely new standard, but the DT document can be merged as a part of DTDL or WoT TD. We propose that the DT document is a YAML file that follows a uniform structure, and we created an example implementation of this file for the digital twin of an overhead crane.

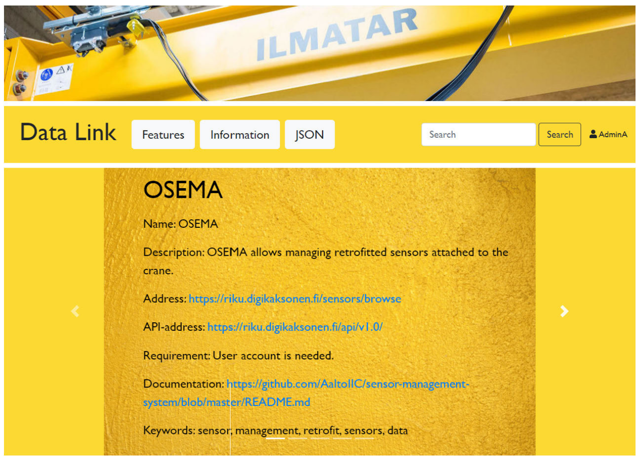
The DT document includes mandatory fields such as standard version, digital twin id, name, and description, and several optional fields such as the location of the physical product, manufacturer, and features of the digital twin. The features are described at a high level containing the name and URL of the service, keywords, and service description. The DT document standard draft is publicly available at GitHub [27], and an excerpt from the document is shown in Fig. 2. DT document does not provide a means for automatic communication between digital twins, and manual work for establishing a connection between digital twins is required. Nevertheless, the API gateway of the Data Link facilitates the communication between digital twins by forwarding messages to requested features and providing authentication.

```yaml
version: "1.0"
privacy: "public"
id: "http://d-t.fi/ilmatar-K16052"
name: "Ilmatar crane"
description: "The documentation of Ilmatar overhead crane"
createdMachine: "1605277810"
createdHuman: "2020-11-13T14:30:10.555Z"
modifiedMachine: "1605624649"
modifiedHuman: "2020-11-17T14:50:49.124Z"
owner: "Aalto-yliopistosäätiö"
contact:
  name: "John Doe"
  email: "john.doe@aalto.fi"
location:
  streetAddress: "Otaniemi"
  gpsCoordinates: "60.1841° N, 24.8301° E"
manufacturer: "Konecranes"
features:
  - name: "OSEMA"
    description: "OSEMA allows managing retrofitted sensors
      attached to the crane."
    address: "https://example.sensor.fi/sensors/browse"
    apiAddress: "https://digi.kaksonen.fi/api/v1.0/"
    requirement: "User account is needed."
    documentation: "https://github.com/AaltoIIC/
      OSEMA/blob/master/Documentation.md"
    keywords:
      - "sensor"
      - "management"
      - "retrofit"
      - "sensors"
      - "data"
  - name: "MindSphere"
```

FIGURE 2. Excerpt from the public part of the DT document describing the metadata of the overhead crane digital twin.

IV. DATA LINK

The Data Link offers a flexible and scalable way of building digital twins, as a digital twin may only implement the necessary features and additional features can be easily added based on need. At simplest, a digital twin may only consist of its metadata description, i.e., DT document. The Data Link has two parts: the user interface that allows a human-friendly way to examine the information and features of the digital twin (Fig. 3) and an API gateway which links the features of a DT behind a single API and facilitates their communication. API gateway forwards requests to cor-



OSEMA

Name: OSEMA

Description: OSEMA allows managing retrofitted sensors attached to the crane.

Address: https://riku.digikaksonen.fi/sensors/browse

API-address: https://riku.digikaksonen.fi/api/v1.0/

Requirement: User account is needed.

Documentation: https://github.com/AaltoIIC/sensor-management-system/blob/master/README.md

Keywords: sensor, management, retrofit, sensors, data

FIGURE 3. The user interface of the Data Link provides information on the Digital Twin and its features.

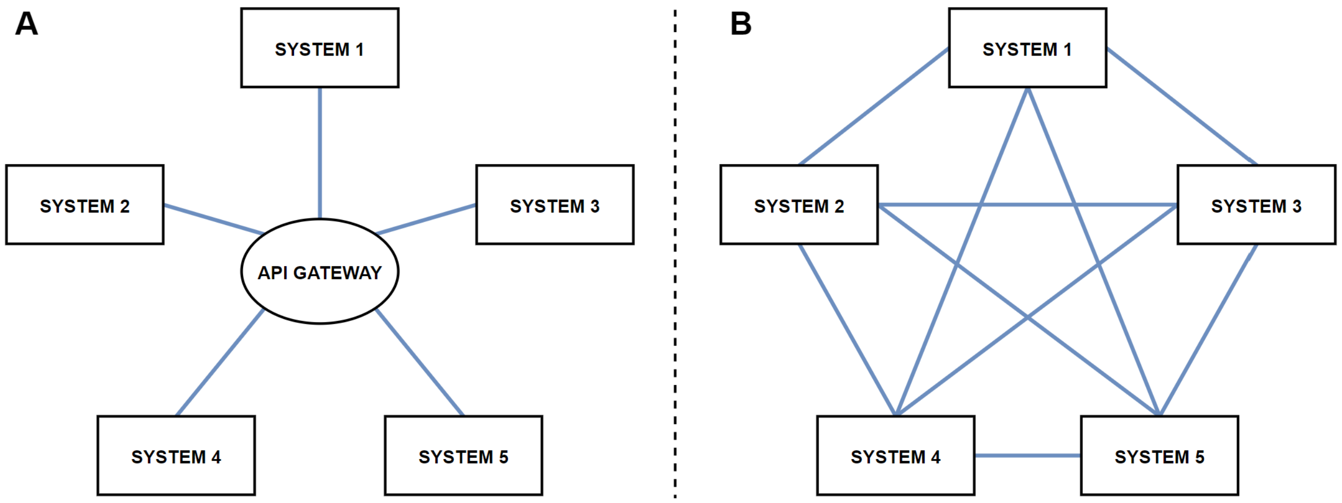


FIGURE 4. Communication with A) Star-like communication is more efficient than with B) Grid style communication [5].

rect services and provides authentication. The API gateway forms a star-like structure to connect the features of a digital twin (Fig. 4a). This structure streamlines the communication by lowering the number of connections per client [5] which can be seen by comparing Fig. 4a and 4b. Thus, a client, i.e., service, needs to know only the URL and credentials of the API gateway. Compared to the situation in which each service needs to store information on how to connect other services, API simplifies access management. For example, if the authentication method or credentials are changed for one service, the corresponding change has to be made only to API gateway, not to all other clients. The more there are services, the more beneficial is the centralized management of authentication. In addition, the API gateway has two main purposes:

1) It provides a single access point to the features of a digital twin. This access point can be used by external services or other digital twins.
2) Services can use other services to implement their functionalities by communicating with one another via the gateway. For example, a predictive maintenance feature can fetch both historical data and simulation models using the gateway and then use them for calculations.

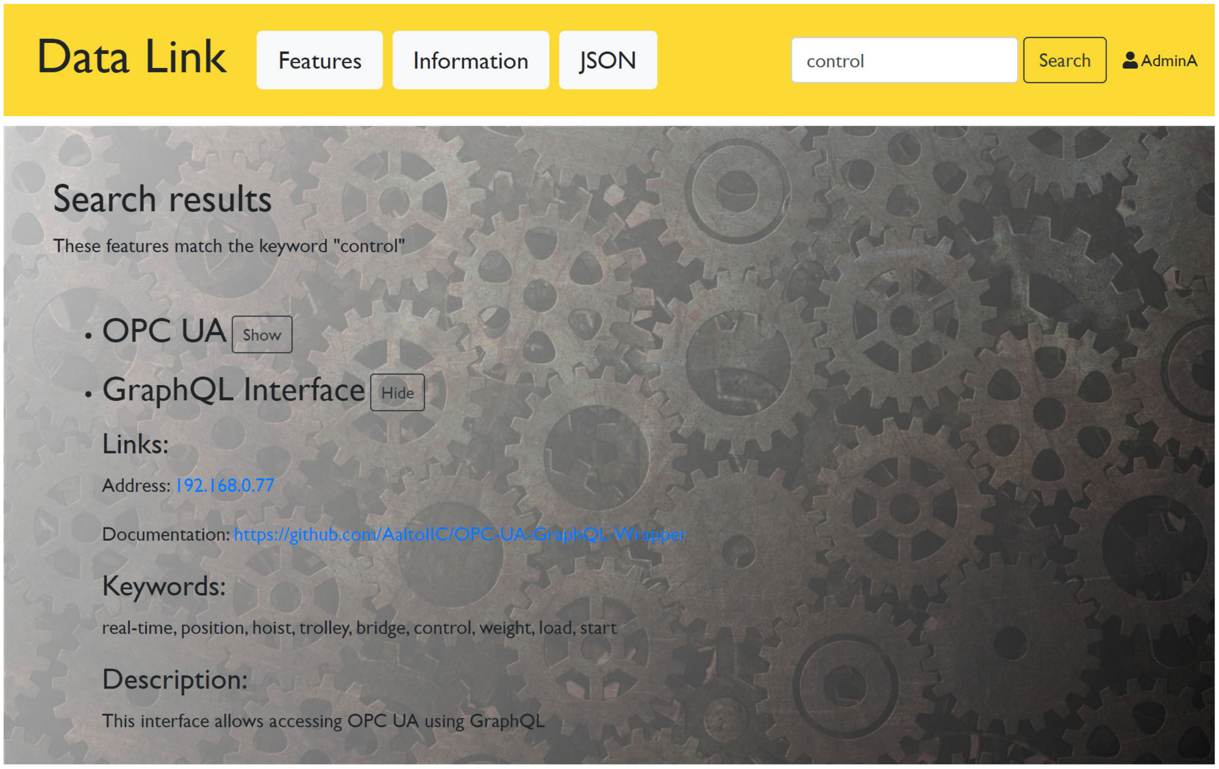
In this paper, Flask, a Python web framework, was used to implement the API gateway. There are also several commercial solutions to API gateways such as Amazon API gateway [28] and Azure API Management [29] available. Writing our own API gateway enabled full-control of the gateway logic, which allowed, for example, implementation of custom authentication methods. The API gateway forwards a request to a service with the following steps:

1) Client sends a request to API gateway to address <gateway url>/services/<service name>/<subpath>, in which service name defines the service and subpath the resource accessed from the target API.
2) API gateway checks if the authentication token sent with the request is valid. If it is not valid, a response with HTTP status code *401 Unauthorized* is returned. In addition, if the user does not have the rights to forward a message, a response with HTTP status code *403 Forbidden* is returned.
3) Based on the service name, information on how to connect to the service such as URL, authentication method, and credentials are fetched from the gateway database. If the service does not exist in the database, a response with status code *404 Not Found* is returned.
4) API replaces the authentication fields and target URL from the original request and forwards the request to the requested service API.
5) If the target service responds with status code *401 Unauthorized*, the gateway continues to the next step. Otherwise, the response is forwarded back to the client, and the message forwarding is completed.
6) API gateway tries to refresh the credentials for the requested service. If the operation is successful, the gateway continues to the next step. If it is not, the initial response from the service API with status code 403 is forwarded back to the client.
7) The request is forwarded to the service API, and the response is returned to the client. The message forwarding is completed.

The user interface provides an overview of a digital twin and its capabilities, i.e., features of the digital twin (Fig. 3). This information is stored in a DT document introduced in the previous section. The UI includes a search function (Fig. 5) that allows finding the correct feature or piece of information for a specific task. Currently, the search function relies on the metadata stored on the DT document. The user interface is a single page application implemented with React. The application back-end was built with Node.js and Express framework, and it uses a MongoDB database.

V. THE IMPLEMENTATION OF THE DATA LINK FOR AN OVERHEAD CRANE

The Data Link was implemented for an overhead crane (Fig. 6), which acts as an example of a large industrial



Search results

These features match the keyword "control"

- OPC UA
- GraphQL Interface

Links:

Address: 192.168.0.77

Documentation: https://github.com/AaltoIIC/OPC-UA-GraphQL-Wrapper

Keywords:

real-time, position, hoist, trolley, bridge, control, weight, load, start

Description:

This interface allows accessing OPC UA using GraphQL

FIGURE 5. Search functionality allows finding services with desired data.

FIGURE 6. Ilmatar the overhead crane for which the digital twin was created.

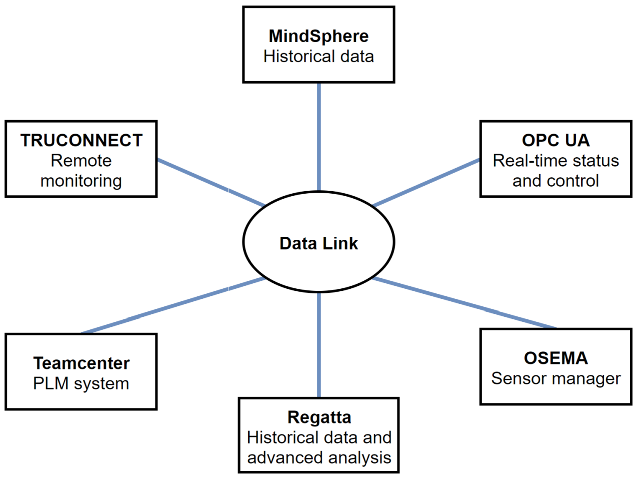


FIGURE 7. Data Link connects the separate systems forming a digital twin of an overhead crane.

machine. The crane has several additional features compared to ordinary cranes in the field, such as MindSphere connectivity, and is not a standard solution from the manufacturer. Currently, several separate systems store the crane data. These systems can be used to implement a subset of the features of the digital twin of the crane. However, not all of these systems have Web APIs, and therefore, they can not be connected to the API gateway directly. In this case, Data Link provides only the metadata about the system to allow at least finding the system with relevant information. The Data Link is visualized in Fig. 7, and the list below describes each system added to the Data Link:

1) Siemens MindSphere stores the historical data about the crane, such as position and distance driven. It offers a web user interface that provides visualizations of the data and an extensive REST API connected to the API gateway. MindSphere allows developing applications on top of it, and we have developed a bearing lifetime estimation application for the crane rope sheaves [30].
2) TRUCONNECT is a service for remote monitoring provided by the manufacturer of the crane. It offers a web user interface that displays the condition of the crane, such as the wear of the brakes. In addition, operation statistics, such as hours driven, and possible

safety alerts, such as emergency stops and overloads, are available in this service. The service helps the user in the interpretation of the statistics and recommends actions to improve the operation of the crane, such as training of operators to avoid emergency stops. TRUCONNECT does not have an API, and, thus, only details on how to access the web user interface is stored on the Data Link.

3) OPC UA server allows controlling and monitoring of the crane. It is directly connected to the PLC system of the crane. Therefore, it provides access to the crane internal values, which can then be read and modified in real-time. For example, the speed of the trolley can be set via the OPC UA server. The crane also has a few internal sensors, such as laser distance sensors for measuring its position, the readings of which can be read from the OPC UA server. The OPC UA server is connected to the API gateway via GraphQL API.
4) Teamcenter is a PLM software that is used to store CAD files of the crane. It offers a graphical user interface but does not have an HTTP/REST API. Nevertheless, some of the CAD files are hosted on a separate server to be accessible via API gateway.
5) Open Sensor Manager allows managing retrofitted sensors [31]. Currently, the position of the trolley and bridge are measured with two distance sensors, and an accelerometer is attached to the hook of the crane. These sensors can be managed via OSEMA. OSEMA also has a REST API that is connected to the API gateway.
6) Regatta IoT Platform platform stores data from retrofitted sensors and allows data analysis. With Regatta, an application that estimates the usage roughness of the crane has been developed [31]. The application uses measurement data from the accelerometer attached to the crane hook to calculate the roughness index. Regatta also provides a wide range of data visualizations. It offers an extensive REST API connected to the API gateway.

VI. EXPERIMENTS ON THE API GATEWAY

The latency added by the API gateway was measured with a test setup consisting of a client, API gateway, a simple HTTP API server, a laptop, and a router (Fig. 8). The client, API gateway, and the test API were run on Raspberry Pi 4, and a switch mirrored traffic to port 8 from which the laptop captured it with Wireshark software. The client made GET and POST requests both directly and via the gateway to the test HTTP API, and the request execution times were recorded. There were two sizes of responses used with GET requests: the first contained 2 value-pairs as JSON and the second on 100 000 value-pairs. These payload sizes were also used with POST requests, whereas the response for a POST request always contained two value-pairs as JSON. Each request was executed 50 times.

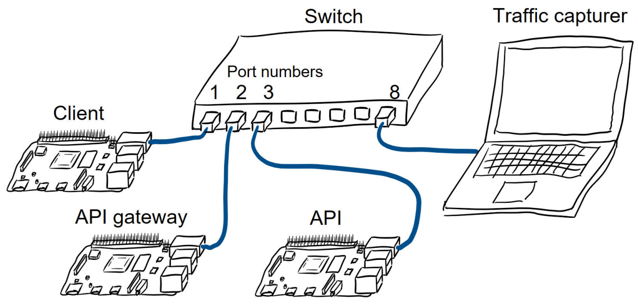


FIGURE 8. **The measurement setup consists of client, API gateway, test API, switch, and laptop capturing network traffic.**

The results indicate that gateway significantly increases the latency (Fig. 9). With a small amount of data sent or received, the latency increases relatively more than with a larger amount of data. The added latency with a small amount of data is approximately 34 ms, whereas, with a large amount of data, it is approximately 120 ms. The statistical properties of the measurements are presented in Table 1.

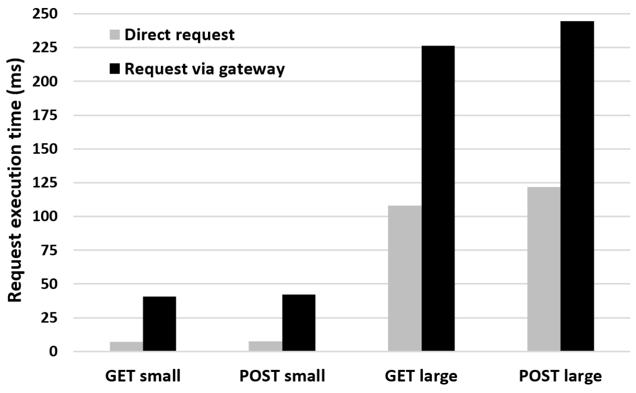


FIGURE 9. **Average request execution times directly and via the API gateway to the test API.**

TABLE 1. **Statistical properties of request execution times. All values are in milliseconds. GW = API gateway, SD = standard deviation.**

Test	Min	Max	Mean	Median	SD
GET small	6.8	8.4	7.1	7.1	0.33
GET small via GW	38.8	47.1	40.5	40.7	1.15
POST small	7.2	8.6	7.5	7.7	0.38
POST small via GW	36.4	47.8	41.8	42.0	1.37
GET large	94.2	115.6	106.2	108.1	5.31
GET large via GW	202.1	245.0	227.3	226.4	8.47
POST large	95.2	142.6	121.2	121.7	12.09
POST large via GW	178.6	285.3	244.4	244.5	18.47

VII. DISCUSSION

A. DATA LINK

Currently, there is no standardized architecture for building digital twins that has led to a wide variety of implementations. This variety makes the interoperability of digital twins challenging. To promote interoperability and clearer structuring of Digital twins, this paper developed further Data Link introduced by Autiosalo *et al.* [5]. The Data Link allows using the existing services to create a digital twin and implement its features. It provides an API gateway for accessing the services, and the services can use the gateway to communicate with one another. In addition, digital twins might use Data Link to communicate with other digital twins.

Building digital twin using independent software blocks or systems allows flexibility since features can be added and removed based on need. In addition, this architecture offers scalability as each block can easily be replaced. For example, if more accurate simulations are needed, the simulation service can be switched to one with more computational power. In this paper, we used an overhead crane as an example, but the architecture is also suitable for both simpler and more complex objects. For example, at simplest, a digital twin may only contain the metadata description, DT document, of itself. On the other hand, a digital twin may consist of several other digital twins, each having various features. For example, a factory-level digital twin may have – in addition to its own features such as a dashboard for the whole factory – references to other digital twins, such as individual production equipment. These references are stored in the DT document of the factory digital twin.

Several papers have already proposed composing a digital twin of several blocks, such as microservices. The Data Link connects these blocks using an ordinary API gateway and enables the communication between the blocks by forwarding requests from one service to another. It lowers the number of connections compared to architectures in which systems communicate directly with each other presented by, for example, Preuveneers *et al.* [9]. Lowering the number of connections simplifies the linking of services because each service does not need to contain information on how to connect to all other services. In addition, the centralized Data Link allows monitoring of communication since all requests go through the gateway.

Numerous commercial and open-source API gateways are available. However, in this paper, the API gateway was implemented by the authors because APIs of the systems implementing features of the overhead crane digital twin used custom methods for updating access tokens. API gateway eases access management since it takes care of the authentication. Thus, services need to authenticate themselves only to the gateway, not to each other, and credentials to services are easier to manage and keep up to date.

The major obstacle for using existing services and systems to implement the features of a digital twin is their lack of interfaces, specifically HTTP APIs. To overcome this, HTTP APIs have to be written to these systems, which might difficult or even impossible due to the closed nature of the systems. If the creation of APIs is not possible, this paper proposes that Data Link contains information on the data these services contain and how to access them. Later, these services might be replaced with new services following microservices architecture, i.e., one service implements one feature or functionality. However, following pure microservices architecture is not necessary for using Data Link, as pointed out in [5], and a single system might implement several features of DT.

To evaluate the suitability of the API gateway for linking the features of a digital twin, the request execution times were measured with the test setup. The results are only extensible to a limited extent since various implementations of API gateways are available. In addition, several other factors affect the overall latency, such as services itself, their location, and network quality. Nevertheless, measurements clearly show that the API gateway adds a considerable amount of latency to request execution times. In the case of small requests, the execution times were increased more than fivefold compared to direct requests. Thus, control and monitoring applications that require low latency should not rely on communication via the Data Link but, instead, communicate directly with the physical product, for example, using its OPC UA server. However, direct communication between services and between services and a physical product would complicate the management of connections, contrary to one of the main motivations for creating Data Link, i.e., simplifying client-side communication. In addition, expect for real-time control and monitoring, almost any other feature can use the API gateway, and new features can be implemented by combining existing features. For example, simulations can be run on models stored on another service, and the results can then be analyzed using yet another service. With features transferring a large amount of data such as a batch of historical data, the significance of request execution time decreases since the whole operation, which often includes heavy data processing, is also more time-consuming. Furthermore, with a large amount of data, the relative difference in execution times between direct requests and requests via API gateway diminishes.

B. DIGITAL TWIN PROTOTYPE FOR AN OVERHEAD CRANE

As the crane for which the Data Link was implemented is used for research and student projects [10], there are often new people unfamiliar with the crane working with it. Therefore, the Data Link that describes the features of the digital twin of the crane is especially useful. For example, if the crane user wants to find the CAD models of the crane, the search functionality can be used (Fig. 5), and information on how to access these files is provided.

The implementation of the Data Link for the crane was straightforward, and the problems were mainly caused by services without HTTP API, i.e., TRUCONNECT and Teamcenter. For TRUCONNECT, we described only how to access it and the data it contains, and for Teamcenter, we hosted the CAD files, which are more static than TRUCONNECT operational data, on the separate server. This underlines the importance of interfaces for constructing uniform digital twin from separate systems. However, all information related to the crane is now accessible from a single endpoint. We have not yet deployed the API gateway to the public Internet because the API gateway needs to be connected to the crane (Wi-Fi) network to be able to communicate with the crane

OPC UA server. In addition, before exposing the gateway to the Internet, the security of the gateway needs to be assured because an attacker capable of physically controlling the crane would cause serious consequences. The quality of the crane network is poor, causing high latency, and thus, the latency added by the API gateway was not considered a major drawback with the crane digital twin.

C. DIGITAL TWIN DOCUMENT

A need to formally describe the digital twin and its features was identified during the development of Data Link. For that, we proposed a DT document, which is a high-level human-readable description of the DT. The document does not yet allow services to communicate with one another automatically, but this will be considered in future development. In order to achieve automatic communication, the interfaces of the services are needed to be documented in more detail and machine-readable format, for example, using a similar approach as WoT TD, in which forms allow communication with the thing interface. However, documenting all interfaces in the DT document would require excessive manual work. Therefore, the DT document currently contains only a link to the original documentation of the service. When automatic communication is achieved, the DT document could allow direct communication between services for low-latency applications. In addition, other digital twins could automatically discover and access digital twin features.

D. FUTURE WORK

Future research includes enabling communication between several digital twins using Data Link and forming an ecosystem containing multiple digital twins. In addition, more intelligent methods for indexing the data of a digital twin could be investigated to allow more advanced search functionality. For wider adoption of the DT document and creation of the Digital Twin Web, an iterative standardization process needs to be initiated. To promote standardization, we have published our proposal for DT document in GitHub [27]. This proposal could be added as a part of Microsoft DTDL or WoT TD standards in the future.

VIII. CONCLUSION

This paper implemented a Data Link following Feature-based Digital Twin Framework architecture presented by Autiosalo *et al.* [5]. The Data Link allows using already existing systems to form a digital twin of an entity. It consists of two parts: the user interface that allows exploring the features of a digital twin and the API gateway that offers a single interface for accessing all information about the physical entity. In addition, the API gateway facilitates the communication of features by forwarding messages between them. The HTTP API interface makes a digital twin accessible on the Internet and allows communication with the DT enabling the interoperability of digital twins. The measurements show that latency added by the API gateway is suitable for connecting the services implementing the features of the digital twin. However, time-critical applications should not communicate via the gateway to minimize latency.

We implemented a prototype digital twin for an overhead crane demonstrating the suitability of the Data Link for creating a digital twin of a large industrial machine. We also identified a need to describe the features of a digital twin and developed a proposal for the DT description document available at [27]. Future research includes standardization of the DT document and creating larger ecosystems consisting of multiple digital twins implemented with the Data Link.

ACKNOWLEDGMENT

Riku Ala-Laurinaho would like to thank Tekniikan edistämissäätiö.

REFERENCES

[1] F. Tao, J. Cheng, Q. Qi, M. Zhang, H. Zhang, and F. Sui, "Digital twin-driven product design, manufacturing and service with big data," *Int. J. Adv. Manuf. Technol.*, vol. 94, nos. 9–12, pp. 3563–3576, 2018, doi: 10.1007/s00170-017-0233-1.

[2] E. Negri, L. Fumagalli, and M. Macchi, "A review of the roles of digital twin in CPS-based production systems," *Procedia Manuf.*, vol. 11, pp. 939–948, 2017, doi: 10.1016/j.promfg.2017.07.198.

[3] S. Boschert and R. Rosen, "Digital twin—The simulation aspect," in *Mechatronic Futures*, P. Hehenberger and D. Bradley, Eds. Cham, Switzerland: Springer, 2016, pp. 59–74, doi: 10.1007/978-3-319-32156-1_5.

[4] H. Laaki, Y. Miche, and K. Tammi, "Prototyping a Digital Twin for Real Time Remote Control Over Mobile Networks: Application of Remote Surgery," *IEEE Access*, vol. 7, p. 20 325–20 336, 2019, doi: 10.1109/ACCESS.2019.2897018.

[5] J. Autiosalo, J. Vepsalainen, R. Viitala, and K. Tammi, "A feature-based framework for structuring industrial digital twins," *IEEE Access*, vol. 8, pp. 1193–1208, 2020, doi: 10.1109/ACCESS.2019.2950507.

[6] K. Borodulin, G. Radchenko, A. Shestakov, L. Sokolinsky, A. Tchernykh, and R. Prodan, "Towards digital twins cloud platform: Microservices and computational workflows to rule a smart factory," in *Proc. the10th Int. Conf. Utility Cloud Comput.*, Austin Texas USA, Dec. 2017, pp. 209–210, doi: 10.1145/3147213.3149234.

[7] D. Taibi, V. Lenarduzzi, and C. Pahl, "Processes, motivations, and issues for migrating to microservices architectures: An empirical investigation," *IEEE Cloud Comput.*, vol. 4, no. 5, pp. 22–32, Sep. 2017, doi: 10.1109/MCC.2017.4250931.

[8] A. B. A. Alaasam, G. Radchenko, and A. Tchernykh, "Stateful stream processing for digital twins: microservice-based kafka stream DSL," in *Proc. Int. Multi-Conf. Eng., Comput. Inf. Sci. (SIBIRCON)*, Novosibirsk, Russia, Oct. 2019, pp. 0804–0809, doi: 10.1109/SIBIRCON48586.2019.8958367.

[9] D. Preuveneers, W. Joosen, and E. Ilie-Zudor, "Robust Digital twin compositions for industry 4.0 smart manufacturing systems," in *Proc. IEEE 22nd Int. Enterprise Distrib. Object Comput. Workshop (EDOCW)*, Stockholm, Sweden, Oct. 2018, pp. 69–78, doi: 10.1109/EDOCW.2018.00021.

[10] J. Autiosalo, "Platform for industrial Internet and digital twin focused education, research, and innovation: Ilmatar the overhead crane," in *Proc. IEEE 4th World Forum Internet Things (WF-IoT)*, Singapore, Feb. 2018, pp. 241–244, doi: 10.1109/WF-IoT.2018.8355217.

[11] M. Grieves and J. Vickers, "Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems," in *Transdisciplinary Perspectives on Complex Systems: New Findings and Approaches*, F.-J. Kahlen, S. Flumerfelt, and A. Alves, Eds. Cham, Switzerland: Springer, 2017, pp. 85–113.

[12] M. Shafto, M. Conroy, R. Doyle, E. Glaessgen, C. Kemp, J. LeMoigne, and L. Wang, "DRAFT Modeling, Simulation, information technology & processing roadmap," Technol. Area, NASA, Washington, DC, USA, Tech. Rep., Nov. 2010. [Online]. Available: https://www.nasa.gov/pdf/501321main_TA11-MSITP-DRAFT-Nov2010-A1.pdf

[13] S. Malakuti, P. van Schalkwyk, B. C. R. Boss; Sastry, V. Runkana, S.-W. Lin, S. Rix, G. Green, K. Baechle, and S. V. Nath, "Digital twins for industrial applications," Ind. Internet Consortium, White Paper, Feb. 2020, pp. 1–19. [Online]. Available: https://www.iiconsortium.org/stay-informed/digital-twins-for-industrial-applications.htm

[14] B. R. Barricelli, E. Casiraghi, and D. Fogli, "A survey on digital twin: Definitions, characteristics, applications, and design implications," *IEEE Access*, vol. 7, pp. 167653–167671, 2019, doi: 10.1109/ACCESS.2019.2953499.
[15] M. Liu, S. Fang, H. Dong, and C. Xu, "Review of digital twin about concepts, technologies, and industrial applications," *J. Manuf. Syst.*, to be published, doi: 10.1016/j.jmsy.2020.06.017.
[16] A. Rasheed, O. San, and T. Kvamsdal, "Digital twin: Values, challenges and enablers from a modeling perspective," *IEEE Access*, vol. 8, pp. 21980–22012, 2020, doi: 10.1109/ACCESS.2020.2970143.
[17] K. Y. H. Lim, P. Zheng, and C.-H. Chen, "A state-of-the-art survey of digital twin: Techniques, engineering product lifecycle management and business innovation perspectives," *J. Intell. Manuf.*, vol. 31, no. 6, pp. 1313–1337, Aug. 2020, doi: 10.1007/s10845-019-01512-w.
[18] M. Mena, J. Criado, L. Iribarne, and A. Corral, "Digital dices: Towards the integration of cyber-physical systems merging the Web of things and microservices," in *Proc. 9th Int. Conf. Model Data Eng.* (Lecture Notes in Computer Science), vol. 11815, K.-D. Schewe and N. K. Singh, Eds. Cham, Switzerland: Springer, 2019, pp. 195–205, doi: 10.1007/978-3-030-32065-2_14.
[19] K. Thramboulidis, D. C. Vachtsevanou, and A. Solanos, "Cyber-physical microservices: An IoT-based framework for manufacturing systems," in *Proc. IEEE Ind. Cyber-Physical Syst. (ICPS)*, May 2018, pp. 232–239, doi: 10.1109/ICPHYS.2018.8387665.
[20] S. Haag and R. Anderl, "Digital twin–Proof of concept," *Manuf. Lett.*, vol. 15, pp. 64–66, Jan. 2018, doi: 10.1016/j.mfglet.2018.02.006.
[21] J. Scheibmeir and Y. Malaiya, "An API development model for digital twins," in *Proc. IEEE 19th Int. Conf. Softw. Qual., Rel. Secur. Companion (QRS-C)*, Sofia, Bulgaria, Jul. 2019, pp. 518–519, doi: 10.1109/QRS-C.2019.00103.
[22] M. Platenius-Mohr, S. Malakuti, S. Grüner, J. Schmitt, and T. Goldschmidt, "File- and API-based interoperability of digital twins by model transformation: An IIoT case study using asset administration shell," *Future Gener. Comput. Syst.*, vol. 113, pp. 94–105, Dec. 2020, doi: 10.1016/j.future.2020.07.004.
[23] M. Ciavotta, M. Alge, S. Menato, D. Rovere, and P. Pedrazzoli, "A microservice-based middleware for the digital factory," *Procedia Manuf.*, vol. 11, pp. 931–938, 2017, doi: 10.1016/j.promfg.2017.07.197.
[24] World Wide Web Consortium (W3C) (2020). *Web of Things (WoT) Architecture*. Accessed: Oct. 30, 2020. [Online]. Available: https://www.w3.org/TR/2020/REC-wot-architecture-20200409/
[25] World Wide Web Consortium (W3C). (2020). *Web of Things (WoT) Thing Description*. Accessed: Oct. 30, 2020. [Online]. Available: https://www.w3.org/TR/2020/REC-wot-thing-description-20200409/
[26] Microsoft. (2020). *Digital Twins Definition Language*. Accessed: Sep. 30, 2020. [Online]. Available: https://github.com/Azure/opendigitaltwins-dtdl
[27] R. Ala-Laurinaho, A. Nikander, and J. Autiosalo. (2020). *Digital Twin Document*. Accessed: Nov. 16, 2020. [Online]. Available: https://github.com/AaltoIIC/dt-document
[28] AmazonWeb Services. (2020). *Amazon API Gateway*. Accessed: Sep. 30, 20220. [Online]. Available: https://aws.amazon.com/api-gateway/
[29] Microsoft. (2020). *API Management*. Accessed: Sep. 30, 2020. [Online]. Available: https://azure.microsoft.com/en-us/services/api-management/
[30] J. Mattila, "Nosturidatan analysointi ja visualisointi IoT-alustalla," B.Sc. thesis, Dept. Mech. Eng., Aalto Univ., Espoo, Finland, 2020.
[31] R. Ala-Laurinaho, J. Autiosalo, and K. Tammi, "Open sensor manager for IIoT," *J. Sensor Actuator Netw.*, vol. 9, no. 2, p. 30, Jun. 2020, doi: 10.3390/jsan9020030.

RIKU ALA-LAURINAHO (Graduate Student Member, IEEE) was born in Espoo, Finland, in 1995. He received the B.Sc. and M.Sc. degrees from Aalto University, in 2018 and 2019, respectively. He is currently pursuing the Ph.D. degree with Aalto University. He was a Research and Teaching Assistant with the Department of Computer Science, where he was involved in the development of learning materials. His current research interests include interoperability of digital twins, digital twin ecosystems, and communication in smart factories.

JUUSO AUTIOSALO (Graduate Student Member, IEEE) received the B.Sc. and M.Sc. degrees from Aalto University, in 2015 and 2017, respectively, where he is currently pursuing the Ph.D. degree. He has instructed four master's theses and two bachelor's theses and performed teaching activities at three university courses.

He was the Project Manager for the DigiTwin project, which was executed in tight collaboration with industry partners. His research interests include the practical implementation and network effects of digital twins. Before focusing on the intersection of the Internet of Things and machine design, he implemented several study projects on mechatronics and wrote his M.Sc. thesis on hydraulic accumulators.

ANNA NIKANDER was born in Helsinki, in 1999. She is currently pursuing the B.Sc. degree in information networks with Aalto University, Espoo, Finland. She is also a Research Assistant with the Mechatronics Group alongside her studies. She is also a Course Assistant with the Department of Computer Science, Aalto University. Her current research interests include new service models enabled by Digital Twin and improving the user experience of industrial internet of things applications with the help of augmented reality.

JOEL MATTILA received the B.Sc. degree from Aalto University, Espoo, Finland, in 2020, where he is currently pursuing the M.Sc. degree in mechanical engineering. He is also a Research Assistant with Aalto University alongside his studies.

KARI TAMMI was born in 1974. He received the M.Sc., Lic.Sc., and D.Sc. degrees from the Helsinki University of Technology, in 1999, 2003, and 2007, respectively. He received a Teacher's pedagogical qualification from the Häme University of Applied Sciences in 2017.

He was a Researcher with the CERN (European Organization for Nuclear Research) from 1997 to 2000, and a Postdoctoral Researcher with North Carolina State University, USA, from 2007 to 2008. From 2000 to 2015, he was a Research Professor, a Research Manager, the Team Leader, and other positions with the VTT Technical Research Centre of Finland. He has been an Associate Professor with Aalto University since 2015. He currently serves as a Chief Engineer Counselor for the Finnish Administrative Supreme Court. He has authored over 90 peer reviewed publications cited in over 5000 other publications. He is also a member of the Finnish Academy of Technology. He also serves as the Deputy Chair for IFTOMM Finland.