Received July 30, 2021, accepted September 25, 2021, date of publication October 13, 2021, date of current version October 21, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3119487

Twinbase: Open-Source Server Software for the Digital Twin Web

JUUSO AUTIOSALO[1,2], JOSHUA SIEGEL[2], AND KARI TAMMI[1]
[1]Department of Mechanical Engineering, Aalto University, 02150 Espoo, Finland
[2]Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824, USA

Corresponding author: Juuso Autiosalo (juuso.autiosalo@aalto.fi)

This work was supported in part by the Business Finland through MACHINAIDE under Grant 3508/31/2019.

ABSTRACT Digital twins are expected to form a network, a ''Digital Twin Web,'' in the future. Digital Twin Web follows a similar structure to the World Wide Web and consists of meta-level digital twins that are described as digital twin description documents and distributed via Digital Twin Web servers. Standards must be established before the Digital Twin Web can be used efficiently, and having an easily accessible server implementation can foster the development of those standards. Twinbase is an open-source, Git-based Digital Twin Web server developed with user-friendliness in mind. Twinbase stores digital twin documents in a Git repository, modifies them with Git workflows, and distributes them to users via a static web server, from which the documents can be accessed via a client library or a regular web browser. A demo server is available at https://dtw.twinbase.org and new server instances can be initialized free-of-charge at GitHub via its browser interface. Twinbase is built with GitHub repository, Pages, and Actions but can be extended to support other providers or self-hosting. We describe the underlying architecture of Twinbase to support the creation of derivative and alternative server implementations. The Digital Twin Web requires permanent, globally accessible, and transferable identifiers to function properly, and to address this issue, we introduce the concept of digital twin identifier registry. Performance measurements showed that the median response times for fetching a digital twin document from Twinbase varied between 0.4 and 1.2 seconds depending on the identifier registry.

INDEX TERMS Cyber-physical systems, cyberspace, digital twins, digital twin web, Internet of Things, internet topology, metadata, metamodeling, open-source software, semantic web, web services.

NOMENCLATURE

AAS	Asset Administration Shell
API	Application Programming Interface
DTDL	Digital Twins Definition Language
CI	Continuous/Content Integration
DT	digital twin
DTID	digital twin identifier
DTW	Digital Twin Web
GADIT	Git-based architecture for digital twins
HTTP(S)	Hypertext Transfer Protocol (Secure)
IoT	Internet of Things
IRI	International Resource Identifier
JSON	JavaScript Object Notation
URL	Uniform Resource Locator
YAML	Yaml Ain't Markup Language
WoT TD	Web of Things Thing Description
WWW	World Wide Web

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

I. INTRODUCTION

Digital twins (DTs) are virtual counterparts for real-world entities. The contents of digital twins vary by use case and application domain. Industrial cases have emphasized simulation-focused tasks such as mirroring [1] the life of space vehicles [2], [3], factory floor planning [4], and product design [5]. Internet of Things researchers started from information management focused digital twins to track physical products with RFID tags [6] with applications in logistics [7] and intelligent control of manufacturing [8]. Buildings are adopting digital twins as well [9], ranging from small-scale sensor experiments [10] to a complex case study of a university campus with an array of data sources, multiple stakeholders, and several different services [11]. Nowadays, digital twins are being made for almost all physical things, including humans [12], [13], cars [14], [15], water systems [16], and ice cream machines [17], and even intangible entities, such as organizations [18].

The twins are often implemented with software not originally made for creating digital twins or with project-specific custom software. Perhaps due to these fragmented origins, digital twins are commonly built as isolated entities rather than as parts of fleets that span across stakeholders. Also numerous security challenges [19]–[21] limit the creation of networked digital twins. Despite the challenges in practical implementation of networked twins, researchers have expressed the vision to create a network of digital twins [22]–[26].

Several standard methods for creating digital twins have been proposed [27], although not all of them were made specifically to create digital twins. Currently, the three most prominent appear to be Web of Things Thing Description by World Wide Web Consortium [28], Digital Twins Definition Language by Microsoft Azure [29], and Asset Administration Shell by Plattform Industrie 4.0 [30]. Other standardization and implementation efforts worth mentioning include ETSI NGSI-LD [31], PADI Connection Profiles [32], and Eclipse Ditto platform [33]. Still, none of the mentioned standards has been accepted as the predominant digital twin standard. A digital twin developer is not able to select one standard when they want to create a networked twin that works across several platforms.

Some of the existing standards also seem inappropriate for use with digital twins due to terminological/philosophical inconsistencies; they seem to use the terms thing and twin interchangeably. A twin is a mirrored representation of a thing, and a twin can include a description of the thing it mirrors, but they are still two separate entities. It is okay for users of digital twins to remain ignorant of this difference, but standards must acknowledge and leverage this distinction. Otherwise, building a global network of digital twins will remain an unconquered challenge.

Stemming from the urge to create a global network of digital twins, our three unsubstantiated yet experience-based claims are: 1) most digital twins should have a publicly available meta-level description, 2) relations between digital twins should mimic the relations between real-world entities, and 3) it should be trivial to initialize a public meta-level digital twin for any real-world entity. These manifest-like claims should provide a fertile ground for the creation of the network of digital twins.

We are calling the global network of digital twins the "Digital Twin Web" (DTW) [26] to emphasize the intended resemblance to the World Wide Web (WWW) [34]. The standards of the DTW should be similarly purpose-specific, openly available, connected to each other, and extendable as the standards of the WWW. As a distinction, the DTW can be built on top of the WWW, whereas the WWW is built on top of the Internet. The foundations for the DTW were laid out in the Feature-based Digital Twin Framework [25] and further design principles along with first implementations were presented by Ala-Laurinaho *et al.* [26]. This article presents components that can be used to build the Digital Twin Web as shown in Fig. 1.

This paper introduces "Twinbase," a prototype platform for managing and distributing meta-level digital twins,

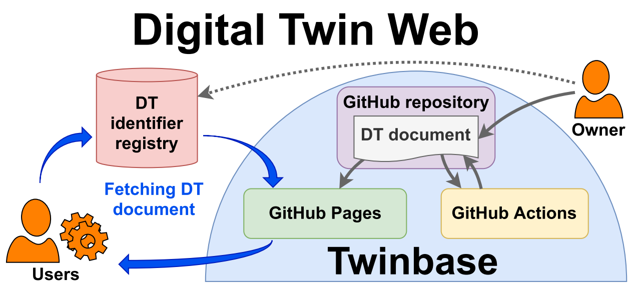


FIGURE 1. The overall workflow for distributing digital twin documents from owners to users with Twinbase and digital twin identifier registry.

i.e., a "Digital Twin Web server." Twinbase stores digital twin description documents in a Git [35] repository and distributes them to human and machine users via a static web server. The twin documents are public by default and can be cross-referenced to each other, rendering Twinbase a suitable tool for building a public network of digital twins. Setting up a new Twinbase instance does not require programming expertise, although a GitHub account is necessary as Twinbase is currently built with three free-of-charge GitHub tools [36]: a public Git repository, GitHub Actions, and GitHub Pages. Using other service providers or self-hosting is possible but not tested and requires expertise in programming and web server administration. Intended future work aims to make Twinbase independent from any specific software provider. Twinbase software is available as open-source from GitHub at https://github.com/twinbase/twinbase with a live demo instance available at https://dtw.twinbase.org.

To support the development of alternative DTW servers, we describe the underlying architecture of Twinbase as generic software architecture, which we call "GADIT" (Git-based architecture for digital twins). GADIT introduces the basic components that Twinbase uses in an implementation-neutral manner. The GADIT architecture requires a separate service that maintains digital twin identifiers that are permanent, globally accessible, and transferable to new owners. To address this need, we introduce the simple yet novel concept of "digital twin identifier (DTID) registry." These registries are thought to become a part of DTW infrastructure, similar to how domain registrars underpin the World Wide Web.

The introduction of the DT identifier registry led to defining the generic structure of DTW, as shown in Figure 2. Along with this structure, this article lays out practices that may become parts of normative DTW standards, such as the specifics of the DT identifier registry, how exactly to find the DT description document, and multiple aspects about the document itself need to be standardized before DTW can flourish. However, the current paper does not attempt to be a normative document but instead concentrates on showcasing Twinbase as a prototype server for the DTW and documenting any observations made during the development. We hope that this prototype server can be used as a tool to develop practices that are mature enough to be set as actual standards.

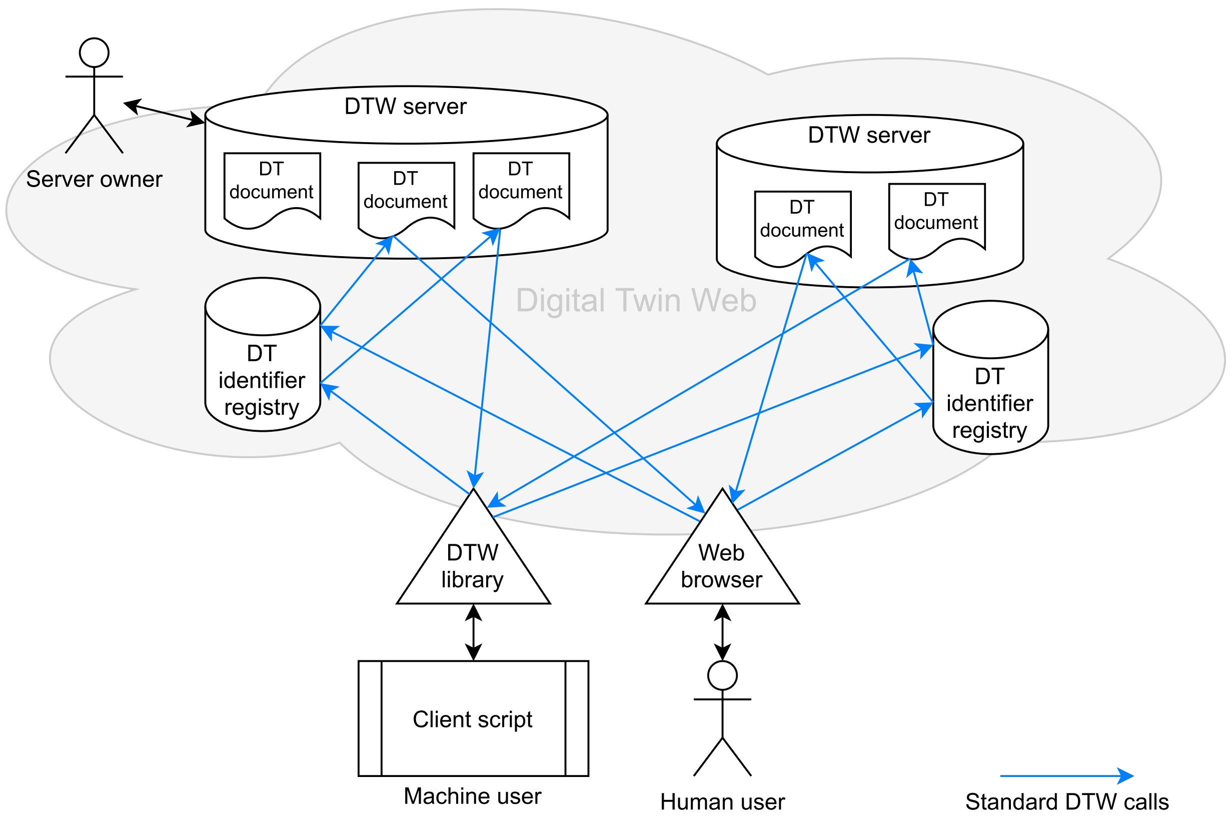


FIGURE 2. The generic structure of the Digital Twin Web. Twinbase can be used as a DTW server. The DT identifier registry is a redirecting database that redirects DT identifiers to DT documents. The DT documents, i.e., the digital twins of the DTW, can be accessed by machine users via a DTW client library and by human users with a regular web browser.

The main contributions of this paper are:

- Introduce the open-source platform Twinbase as the first implementation of a Digital Twin Web server. (Presented in Section IV.) Source code available from GitHub: https://github.com/twinbase/twinbase
- Conduct performance measurements of Twinbase to provide the first estimations of its applicability for the Digital Twin Web. (Section V)
- Refine the structure and design principles of the Digital Twin Web. (Figure 2 and Section VI-D)
- Introduce the basic properties of DTID registries to enable the allocation of permanent, globally accessible, and transferable DT identifiers. (Section III-F)
- Introduce GADIT (Git-based architecture for digital twins) as a general architecture for implementing Git-based servers for the Digital Twin Web. (Section III).

The remainder of the paper is organized as follows. Section I-A describes the motivation for creating Twinbase and Section II presents related work. Section III introduces GADIT as the underlying architecture of Twinbase, and Section IV presents how Twinbase was implemented. Section V shows the results of performance measurements and Section VI discusses Twinbase from several viewpoints. Section VII describes risks and limitations, Section VIII summarizes the managerial implications of the work, and Section IX concludes the article.

A. MOTIVATION

Despite the mass adoption of digital twins in recent years, there are practically no digital twins online and ready to be browsed on the Internet. The absence is peculiar as digital twins are supposed to be heavily connected entities and, e.g., the Gemini Principles of the Centre for Digital Built Britain mention "public good" and "openness" as two of nine basic principles [37]. The centre hosts a DT Hub whose first annual report on year 2020 mentioned "openness" as one of four ideas to improve and "Practical digital twin examples" as one of four recommendations for next year in its community response appendix [38]. A recent article written by representatives from seven commercial organizations emphasizes the role of open standards and open-source software in facilitating the adoption of digital twins [39]. In essence, the need for open digital twins has been recognized, but implementation is not sufficient.

We surmise that the lack of internet-accessible twins stems from three main reasons. First, implementations of the digital twin concept were first made in locally run software, and making twins accessible via internet platforms has been an afterthought that has not yet caught on properly. Hence, there is an apparent lack of tools for building internet-accessible twins. Second, most digital twins contain confidential information, and as most tools do not enable sharing twins only partially, a culture of sharing digital twins has not been formed. Third, the demand for the creation of digital twins comes from practitioners that usually do not have the programming skills to create internet-accessible DTs. It also seems that many DT practitioners even lack the basic understanding of why DTs should be accessible over the Internet, although this may be a fallacy due to underdeveloped tool offerings. Nevertheless, it seems that DT creators do not have either motivation or skills to create internet-based digital twins.

Reaching a wider adoption rate of internet-based DTs will require that they are much easier to create in several aspects. In an optimal situation, the purpose of creating internet-based digital twins is self-evident, their creation is instant, they cost no money, require no maintenance, and do not create vendor lock-in. (On the contrary, DTs create value by opening multiple connections and using multi-vendor software offerings.) And all of these requirements should apply even to non-experts. Also, reliable and affordable support should be readily available in case anything goes wrong. If taken as absolutes, these are impossible requirements, but when given some leeway, they in fact describe many existing, proven technologies, and the physical world is full of these types of basic products. For example, bolts, power cords, and light bulbs are practically ubiquitous, meaning several different companies manufacture them, and they are available at a wide array of stores. Of course, they do cost money due to, e.g., material, manufacturing, and logistics costs, but development costs for these basic products are practically negligible, and competition keeps prices low.

Digital products have no material, manufacturing, or logistics costs, so development costs play a decisive role in the price of digital products. When using freely licensed open-source products, the basic products of the digital world, users don't pay for development costs directly. However, digital products are often used and offered as services, which creates server upkeep costs. We believe that the networked digital twins of the Digital Twin Web will become such service-based basic products. Hence, the creation and upkeep of networked digital twins will include two main cost categories: development and server upkeep costs. Additionally, there are DT content updating costs, but they differ by use case and require their own cost analysis and software solutions.

Development costs can be lowered significantly if DTs can be created with open-source software. We assume that not all the sophisticated features of DTs will be available as open source, but the basic core service of DTs will become open source. We expect it to be comparable to the free WWW servers, such as Apache [40] and Nginx [41]. Twinbase is our suggestion for an open-source DTW server implementation.

The server upkeep costs of DTs depend on how computationally intensive the server implementation is. If a DT needs a whole server to itself, costs are high, typically around $10 per month. However, if several DTs can be served from a single static web server, costs go down remarkably. Some providers (e.g. GitHub Pages [42] and Netlify [43]) even offer static web servers free-of-charge. Twinbase is built with these free-of-charge services to keep server upkeep costs at a minimum.

II. RELATED WORK

To the authors' best knowledge, there is no other architecture or implementation that allows serving meta-level digital twins from a static web server that fetches its contents from a Git repository. However, there are services that allow the distribution of meta-level digital twins in other means.

Azure Digital Twins [44] is a cloud platform for creating internet-based digital twins offered by Microsoft. It leverages an open-source Digital Twins Definition Language (DTDL) [29], also developed by Microsoft, and includes other open source components, such as a visual explorer for browsing a network of digital twins called Azure Digital Twins Explorer [45]. The core of Azure Digital Twins is closed source and offered as a pay-per-use service. Azure Digital Twins has been used as a platform to build other commercial services, such as e-Magic TwinWorX [46] and MindSphere City Graph [47]. It remains to be seen if the partly-open, partly-closed model of Azure Digital Twins will be a successful method for building an ecosystem of networked twins.

Asset Administration Shell (AAS) is the implementation of digital twin by the Plattform Industrie 4.0 [30]. AAS is used to describe the basic information of industrial products similarly to the DT document. There are open-source software available for viewing and editing AAS [48] as well as for serving them over the internet [49]. AAS models are defined in the technology-neutral UML language, and there are mappings to OPC UA, XML, JSON, RDF, and AutomationML. At the time of the writing, there was also a demo server available at http://www.i40-aas.de showing AAS information for 33 devices from 14 different manufacturers. The drawbacks of AAS are the exclusivity to industrial equipment and that it seems to offer no clear method for connecting twin instances to each other, hence not being an ideal tool for creating general networks of digital twins. Nevertheless, a substantial amount of work has clearly been done for AAS, and it seems especially suitable for describing various technical aspects of industrial devices.

Web of Things Thing Description (WoT TD) is a standard information model for describing things, developed by the World Wide Web Consortium [28]. WoT TD is formatted as JSON-LD by default and includes the metadata and interfaces of a thing. A WoT TD can represent a physical or virtual entity, although an IoT device such as a lamp seems to be the most common use case. Web of Things and its Thing Description has attracted both open and closed source server implementations. Active open-source projects include Thingweb [50], a WoT implementation written in Node.js with a REST interface, a web user interface, and a validator playground, Eclipse ediTDor [51], a web-based editor that assists in writing WoT TDs, and WoTPy [52], a Python implementation for WoT. Regarding closed source implementations, Philips Hue and Azure IoT has been said to

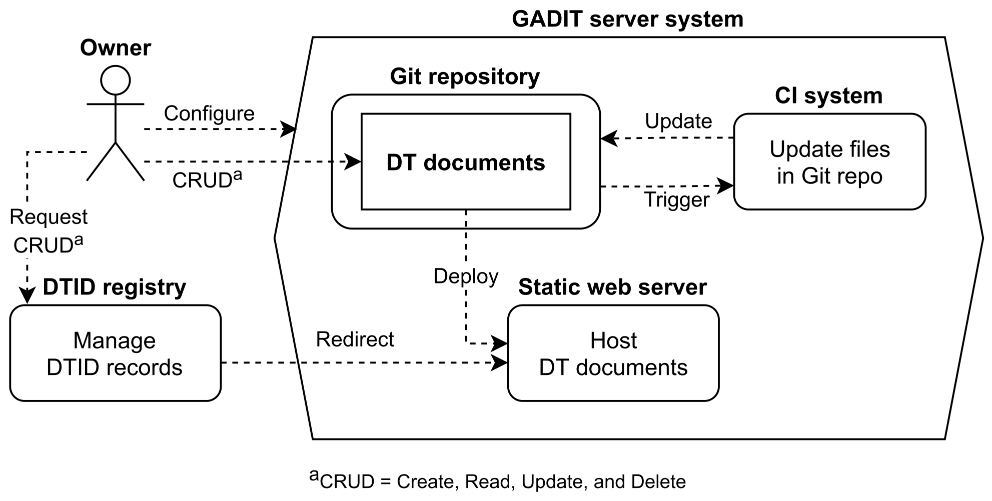


FIGURE 3. A GADIT server system consists of a Git repository, a CI (Content Integration) system, and a static web server. The Git repository contains DT documents and other files, which are updated by the CI system and deployed to the static web server for distribution. The owner and the DTID registry are not parts of the server, but they are required for the platform to function. The owner configures the GADIT server and performs CRUD operations on DT documents and DTID records. The DTID registry manages the records and redirects DTIDs to the corresponding twins on the static web server.

leverage WoT TDs [53], and EVRYTHNG [54] has provided a commercial Web of Things platform since 2011, although it is unclear if it leverages TDs. WoT TD is developed inside the WWW Consortium, following the same proven methods that gave us the web, and especially openness of decision-making is at an excellent level. The downsides of WoT TD are that it is focused on things instead of twins, making its positioning unclear, and that the community seems to be programmer-heavy, creating a language barrier between digital twin engineers that concentrate on mirroring complex physical phenomena.

All of the reviewed meta-level digital twin solutions have attracted a noticeable amount of users and developers, but none of them has become similarly ubiquitous as the WWW standards. Therefore, exploring different solutions is necessary until a common approach for building the Digital Twin Web is found. This article attempts to provide a straightforward approach for distributing meta-level digital twins across the Internet while leaving enough leeway so that further work can combine it with other existing solutions.

The DTID registry has functionalities that resemble some earlier work, such as the Auto-ID system [8] and the DIALOG system [7]. When those were developed, physical ID readers were scarce, and the systems did not reach wide consumer outreach. Nowadays, smartphones with QR-code readers are practically ubiquitous in developed countries, which means that a digital twin identifier should be readable with those to reach consumer acceptance. The IDs of Auto-ID and DIALOG systems do not direct users to any additional information with regular QR-code scanner software, which means that they are not useful with current technology. In contrast, the Digital Link standard by GS1 [55] embraces usability with smartphones. The standard is still partly under development, and our plan is to synchronize the DTID registry concept with it when details are fixed.

III. ARCHITECTURE: GADIT

GADIT (Git-based architecture for digital twins) is a general architecture for implementing Digital Twin Web servers using the Git version control system and its common peripherals. The primary purpose of a GADIT server, as well as any other DTW server, is to store, manage, and distribute DT documents. The main components of GADIT are shown in Figure 3 and described in Sections III-A–III-F. GADIT was used to build Twinbase (Section IV) and can be used for building alternative implementations.

GADIT follows the conceptual aspects of the Feature-based Digital Twin Framework [25] and promotes ease-of-use and openness as guiding principles as deemed appropriate by Autiosalo *et al.* [56]. GADIT also adheres to the basic design practices of the Digital Twin Web described in Section III of Ala-Laurinaho *et al.* [26].

A. OWNER

Owner is a legal person who owns three types of entities defined by GADIT: the server itself, the contents of digital twin documents, and the identities of the digital twins. The ownership of the server and the contents are defined by the ownership of the Git repository (Section III-C) and the ownership of the identities is recorded in a DTID registry (Section III-F). The owner as a real entity is not a part of the technical system, but being able to clearly manage the ownership of the digital twins is a key feature of the architecture. Additionally, the meta-level digital twins made with GADIT may link to external digital twin components whose ownership is defined by the respective external system.

Proper management of ownership keeps the legal situation and responsibilities of different components of digital twins clear. If one owner owns all components, the situation is simple, but if, e.g., the identifier and the document are owned by different legal entities, there is potential for conflict, rendering a usage agreement appropriate.

B. CONTENT: DIGITAL TWIN DOCUMENT

DTW needs a way to describe the contents of the digital twins, and we claim that the contents should be described in a digital twin document similarly as the basic content of a web page is described by an HTML document. We define a digital twin document as a text file that provides a meta-level description of a given digital twin. The document can directly describe various aspects of a twin or provide pointers to external components of the digital twin in the spirit of the ‘‘Data Link’’ feature introduced by Autiosalo *et al.* [25]. A DT document is considered to be the master metadata source of a digital twin, so if something is not mentioned in the twin document, it is not considered a part of the twin. In GADIT, twin documents are self-sufficient descriptions and can be moved as such to another DTW server.

Digital twin document does not yet have a single commonly agreed implementation format. Globally acknowledged prospects include Digital Twin Description Language (DTDL) [29], Web of Things Thing Description (Wot TD) [28], Asset Administration Shell (AAS) [30], and other standards investigated by Jacoby and Usländer [27]. However, each of these is already a complex standard, and leveraging them currently requires going through a lot of documentation which does not suit well the ease-of-use principle of the DTW. It also seems that each of the existing standards is not compatible with the other components of GADIT, which means that a custom document format is currently required. The authors are participating in the development of the DT document [26], [57], which can be modified to accommodate any features required by GADIT. For example, GADIT recommends that the twin identifier (Section III-F), the hosting URL, and the owner of the digital twin is included in the DT document. Also, a document editing URL and a contact method are welcome additions. From a long-term perspective, we believe and suggest that the various features of digital twin description formats start converging and are merged with each other.

C. STORAGE: GIT REPOSITORY

Git is an open-source distributed version control software that especially supports software source code management. Git repository is a version-controlled file storage space that can be used with Git clients. An easy way to set up an internet-accessible Git repository is to use a Git service provider, such as GitHub or GitLab. They provide web interfaces to the repositories and offer both public and private repositories free-of-charge at the time of the writing.

In GADIT architecture, a Git repository is used to store both the DT documents of each DTW server instance and the software code of the DTW server solution. The software code consists of back-end code for continuous integration (Section III-D) and front-end code for the static web server (Section III-E). The Git repository can be copied (or ‘‘forked’’ in Git terminology) by other users to create a new instance of the platform, granted that the license terms allow copying.

The main advantages of using Git as the base technology of a DTW server solution are:

- Files in a Git repository are version controlled by default, which means it is possible to view the DT document at any point in time, and the user who makes a change to the DT document is recorded.
- Git is open source and freely licensed, which means
  1) it can be downloaded easily and free-of-charge, enabling the same for the DTW servers,
  2) the interface definitions are open, so it is possible to create additional customized clients independently,
  3) longevity is guaranteed independently from original developers, and
  4) if deemed necessary at some point, it is possible to independently create a GADIT-specific version of Git.
- Git is a well-established system with a wide community of users in the software development domain. New features are still being added, and they seem to be going into directions favorable for GADIT, such as partial clones [58].
- Git has attracted the integration of two common peripherals: continuous integration workflows and automatic deployment to static web servers. These are essential components of GADIT, and their use is described in detail in Sections III-D and III-E.
- Git servers include their own access management solutions, which means that a GADIT-based DTW server doesn’t require its own access management solutions even if it is hosted completely in the cloud.

D. CODE EXECUTION: CI SYSTEM

Continuous integration (CI) is a practice in which software source code is integrated into the mainline code repository continuously at the same pace as the developers are writing it. Confusingly, CI also refers to a software solution, such as Travis CI [59], GitHub Actions [60], or GitLab CI/CD [61], that enables running tests and other actions for the modified code during the general integration process. In this article, we use CI to refer to the software solution.

In practice, CI is implemented by a configuration file and scripts that together define a CI workflow. CI workflows can be triggered in various ways, such as by a push to the repository or as time-based periodical runs. The workflows may be run on Git provider servers or on separate runner servers. Currently, different runner solutions use at least partly different syntaxes to define the workflows, and to promote interchangeability, a Common Workflow Language [62] is under development. It is common practice to store the workflow files in the respective Git repository.

The GADIT architecture uses CI to automate tasks in three main purposes: configuring general settings, reacting to user actions, and executing tests. Configuration workflows for general settings can, e.g., detect the hosting URL and the owner of the platform automatically and distribute them to the whole platform and all DT documents. User actions, such as edits to DT documents, trigger a workflow that implements further changes to the respective document and to other files if necessary. Tests ensure the platform is working properly. For example, tests can validate the conformity to the DT document standard and ensure the correct redirection of DTIDs.

E. DISTRIBUTION: STATIC WEB SERVER

Static web server is a web server that sends existing files from their file system to clients such as web browsers. (In contrast, dynamic web servers construct custom files each time they receive a request from a client.) Static web servers can be connected to a Git repository so that the files from the repository are deployed to the file system of the server, making the repository contents available in a more user-friendly format. Git providers offer repository-connected static web servers as integrated parts of their platforms (e.g., GitHub Pages and GitLab Pages), but repository owners can also use separate web servers hosted by a wide variety of cloud providers (e.g., Netlify or Amazon S3) or even self-host on their own hardware with open-source software (e.g., Apache or Nginx).

GADIT leverages a static web server for both human and machine interfaces. Firstly, the web server provides a normal browser-readable interface, featuring a dedicated ‘‘DT page’’ for each digital twin hosted on the platform. A DT page shows the contents of a DT document and may even leverage browser-side scripts to display dynamic data from appropriately documented external DT components. Secondly, the static web server is used as a ‘‘static API server’’ [63], meaning it also provides machine-readable DT documents to software clients. The machine-readable DT document must be accessible in a standard filename in a standard location, similarly as current web servers have agreed to send the ‘‘index.html’’ file if the requested URL points to a folder.

In the future, it may be useful to replace the static web server with a dynamic web server to reduce the time it takes from editing a DT document to it being accessible on the server. However, it is important to ensure that the master version of the DT document remains in the Git repository and that the editing history remains useful.

F. IDENTIFIERS: DIGITAL TWIN IDENTIFIER (DTID) REGISTRY

Digital twin identifier (DTID) is a special identifier type that is used to identify digital twins, similarly as IRIs are used to identify any kind of web resources. DTID registry is a functional database that keeps a record of DTIDs, their respective hosting IRIs, and their owners. DTID registries are comparable to the domain name registrars of the WWW as they redirect DTIDs to hosting IRIs and keep track of the ownership of DTIDs. DTID registries are operated by entities that are neutral in respect to the owners of the DTIDs. DTID registry is not a part of a GADIT platform but a separate service required for the platform to function to the full extent. In fact, the DTID registry is rather a part of the Digital Twin Web architecture and described as part of GADIT only because the DTW architecture is not yet defined.

The purpose of DTID registries is to provide permanent, globally accessible, and transferable identifiers for digital twins. We elaborate on these three requirements in the following three paragraphs.

Permanence means that a DTID stays as the identifier of a digital twin and its real-world counterpart indefinitely, unaffected by outside events. A permanent identifier allows referring to a physical thing throughout its lifetime, enabling a comprehensive record of its history, which is one of the key conceptual requirements for a digital twin. The primary mechanism of ensuring permanence is to use an identifier whose only functionality is to be a DTID. The functional separation allows freedom in defining other features of a DTW server, such as the folder structure, as it might be convenient to reorganize the DTs in ways that better support the new company product portfolio. Even the domain name of a DTW server might change in case of company mergers. DTIDs must survive these unexpected events because of how they are used: a common use case for a DTID is to attach it physically into a device where it stays for decades. A simple way to guarantee permanence is to use a separate neutral and trusted DTID registry.

Global accessibility means that the DTID can be used to query the corresponding hosting IRI from anywhere around the Internet, although the query may require authentication if deemed necessary. Accessibility also means that the ID itself is in such form that people and commonly used software can intuitively understand how to use the ID. Intuitive understanding is important because a DTID located in the physical world should be recognized as an interface to a digital twin. Examples of current commonly understood globally accessible identifiers include IRIs that start with a common protocol in the scheme part (e.g., https://orcid.org/0000-0003-3714-748X) or at least use a recognizable domain from the Domain Name System (e.g., orcid.org/0000-0003-3714-748X) to define the hostname. The ID needs to be recognizable also in the physical world, where a QR-code seems to currently be the most prominent method of ensuring overall accessibility.

Transferability of a DTID means that it is possible to transfer the ownership of the DTID to another legal person. The ownership needs to be changed when a physical product is sold to ensure the new owner's proper control of the product. Transferability is especially important if the product has a physically hardcoded DTID, as the new owner should be able to control where that DTID points to, typically to their own digital twin of the device. And even if a physical ID is not used, external links to the digital twin of the physical product should stay intact regardless of the owner. To ensure transferability, the DTID itself should not contain any information

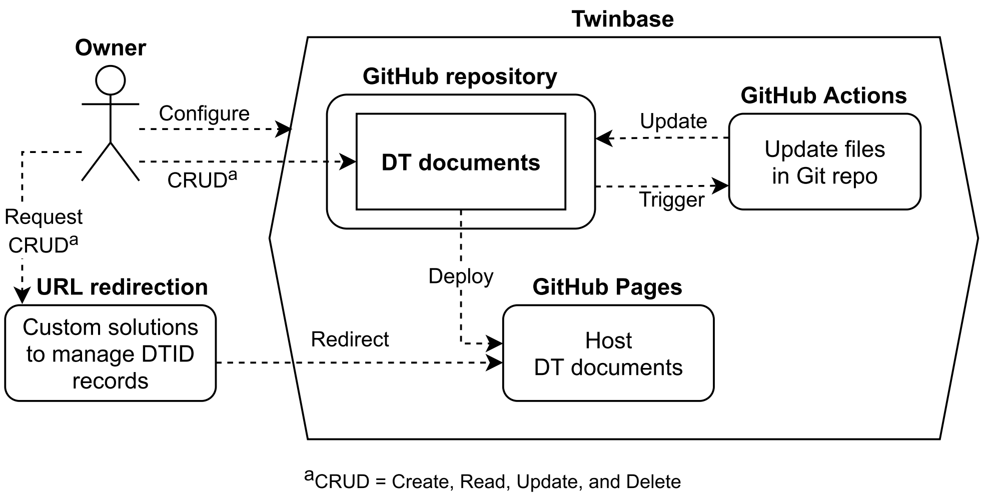


FIGURE 4. Twinbase consists of the same components as the GADIT architecture, implemented with three GitHub services: GitHub repository, GitHub Actions, and GitHub Pages. URL redirection services are used to achieve the functionality of DTID registries.

that identifies any current owner, and ownership is instead defined solely by the record stored in the DTID registry. Hence, the DTID registry needs to be trusted by both the seller and the buyer.

In the future, DTID registries may develop services not mentioned here. For example, the DTID record may be fetched from a standard location in the owner's DT document. It might also be possible that a DTID registry enables the use of a DTID also as a hosting IRI if each DT is assigned its own subdomain, although there may be technical limitations or conventions that limit this practice.

DTID registries already share many similarities with the Domain Name System (DNS), although they point to digital twins instead of IP addresses. As the DNS clearly works very well, future research should find out if DTIDs should be resolved even more with the same methods that the DNS uses. For example, a DTID registry could have several record types for different purposes, similarly to DNS records [64].

As related work, Azure Digital Twins uses Digital Twin Model Identifiers [65] and Asset Administration Shells are required to have a unique identifier [30]. Both support IRI/URL identifiers, but the documentation is unclear, and it seems that the three requirements "permanent, globally accessible, and transferable" are not met by default.

IV. IMPLEMENTATION: TWINBASE

Twinbase is our open-source implementation of the GADIT architecture, distributing digital twin documents to users via a Git repository. The source code of Twinbase is available from GitHub at https://github.com/twinbase/twinbase and licensed under the permissive MIT License. The source code can be forked or used as a template to create a new Twinbase instance that can be hosted directly on GitHub's free-of-charge services. A demo server can be viewed at https://dtw.twinbase.org. The following two subsections present the components (Section IV-A) and user interfaces (Section IV-B) of Twinbase.

A. COMPONENTS OF TWINBASE

Twinbase implements the components of GADIT described earlier in Sections III-A to III-F. The components and their relations are shown in Fig. 4.

1) OWNER

The Twinbase server and the DT documents stored are owned and controlled by the owner of the respective GitHub repository. The owner of the DTIDs of the twins is defined by the DTID registry, which is problematic as proper DTID registries do not yet exist. However, there are two URL redirecting services that can be considered to record ownership and enable ownership change adequately. The w3id.org service seems to be recording the ownership of the DTIDs through a community-based effort, and dtid.org is offered by the authors as a supposedly neutral DTID registry.

2) DIGITAL TWIN DESCRIPTION DOCUMENT: DT DOCUMENT STANDARD DRAFT DEVELOPED AT AALTO UNIVERSITY

Twinbase uses the DT document format introduced by Ala-Laurinaho *et al.* [26]. This format was selected for two main reasons. The authors have the possibility to modify it according to the special requirements that appear during Twinbase development, and the YAML format [66] used by the DT document suits the ease-of-use design philosophy of DTW, making the twin documents approachable with the current human user interface solution of Twinbase.

The format of the twin document is developed as a living standard in GitHub [57]. The format is at an alpha phase and may change substantially in the future. Our intention is to leverage existing standards as much as possible. For example, the WoT TD, DTDL, AAS, or other potential standards identified by Jacoby and Usländer [27] may be leveraged as part of the DT document, or the proven features of the DT document standard may be integrated as part of those. However, it is possible that especially the IoT-rooted standards cannot be used as a base format for digital twins because of the simple distinction described earlier: twins and things are different entities, and this distinction needs to be leveraged in the DT document.

3) GIT REPOSITORY: GitHub

Each Twinbase instance uses a Git repository hosted by GitHub to store its source code and DT documents. The Twinbase template repository can be duplicated either with the "Use this template" feature of GitHub or by creating a fork of the repository. The recommended method for updating Twinbase is not yet defined, as more experimentation is required to define a robust workflow that ensures user-friendly update procedures. Twinbase uses public repositories by default, but customized solutions with private repositories can also be built.

The user-facing contents of Twinbase, i.e., the files that are distributed with the static web server, are stored in the `docs/` folder of the repository. The DT documents are located in their own folders to keep the twins organized and to give a simple hosting URL for the twins. Users can modify twin documents directly in the browser interface of the GitHub repository.

The advantages of using GitHub as the Git repository provider of Twinbase are:

- Free-of-charge services that are suitable for implementing GADIT.
- Widely recognized service with a large user base.
- User-friendly features.

The main disadvantage of using GitHub is that GitHub itself is closed source, rendering Twinbase unusable if GitHub decides to shut down a service that Twinbase uses.

4) CI SYSTEM: GitHub ACTIONS

Twinbase uses GitHub Actions to execute CI workflows that configure general settings, react to user actions, and perform tests as defined earlier in Section III-D. The Actions are defined in the `file-modifier.yml` file in the `.github/workflows/` folder, which utilizes some longer scripts stored in the `.github/` folder. The file-modifier.yml workflow is triggered every time something is pushed into the repository.

Twinbase workflows include several functionalities whose details can be reviewed from the source code. The essential ones are:

- Modifying DT documents, e.g., adding an editing URL.
- Converting YAML document to JSON document.
- Generating a base YAML file with a list of hosted twins.
- Copying index.html file to each new twin folder.
- Testing if DTIDs redirect to hosting URLs.

5) STATIC WEB SERVER: GitHub PAGES

Twinbase uses GitHub Pages as a static web server that hosts the user interface defined in the `/docs` folder of the repository. By default, Twinbase is hosted from URL in the form https://<GitHub username>.github.io/<repository name>, but customized domains are also supported via GitHub Pages settings. When creating a new Twinbase, the Pages need to be enabled from repository settings manually. The user interfaces hosted by GitHub Pages are further described in Section IV-B.

6) DTID REGISTRY: URL FORWARDING SERVICES

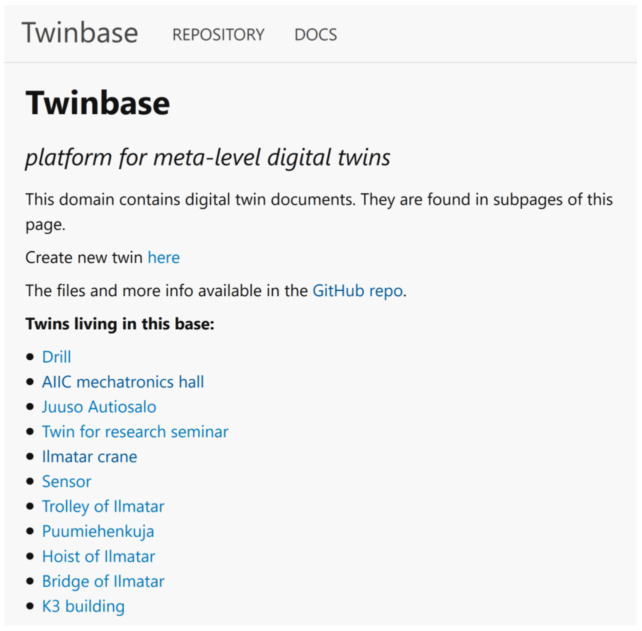
As actual DTID registries do not yet exist, various URL redirection services were used while developing Twinbase. The basic technical functionality of a DTID registry is achieved by redirecting DTIDs to the corresponding Twinbase hosting URLs with HTTP status code 301 or 302. On top of this, a DTID registry needs to manage the ownership of the DTID, and on the owner's request, modify the DTID record or transfer its ownership.

DTID registry is not a component of Twinbase, but the twins of Twinbase rely on DTIDs. Each DT document includes its DTID, and the relations between the twins of Twinbase instances are made through DTIDs. To ensure that all twins are functioning properly, Twinbase checks if the DTID mentioned in a DT document redirects to the hosting URL of that document with the mentioned HTTP status codes 301 or 302.

The main difficulty of using general URL redirection services as DTID registries is that the ownership of the redirected IRIs is not properly defined. Also, the permanence of the redirection is not guaranteed in the level of reliability that is preferred for twin instances of DTW, and in many redirection services, the target address cannot be changed. However, there are two redirecting services that can be considered to fulfill the requirements, one as a general service and one upkept by the authors.

The Perma-ID service [67] hosted by the W3C Permanent Identifier Community Group offers free-of-charge URL redirection from the w3id.org domain. The redirections are managed via a GitHub repository with maintainers from several organizations that pledge to ensure the continuation of the service. Anyone, including anyone outside those organizations, can add redirections to the service but are required to leave a contact method. The approval of that contact person is required for modifying a redirection before a practice maintainer approves the change, even though this practice is not explicitly stated in the instructions. Hence, the identifiers have owners, and the ownerships seem to be transferable. The identifiers are also claimed to be permanent, which seems justified with the governance model. The Perma-ID service seems to fulfill the requirements for a DTID registry, but as the identifiers are manually handled, the service is likely not suitable for users that need hundreds of identifiers.

The authors set up a DTID registry at dtid.org domain with the Rebrandly [68] service because Rebrandly provides



Twinbase REPOSITORY DOCS

Twinbase

platform for meta-level digital twins

This domain contains digital twin documents. They are found in subpages of this page.

Create new twin here

The files and more info available in the GitHub repo.

Twins living in this base:

- Drill
- AIIC mechatronics hall
- Juuso Autiosalo
- Twin for research seminar
- Ilmatar crane
- Sensor
- Trolley of Ilmatar
- Puumiehenkuja
- Hoist of Ilmatar
- Bridge of Ilmatar
- K3 building

FIGURE 5. Twinbase front page with a list of hosted twins at the bottom. New twins are automatically added to the page. Blue text indicates a clickable link. The interface could be viewed at https://dtw.twinbase.org at the time of writing.

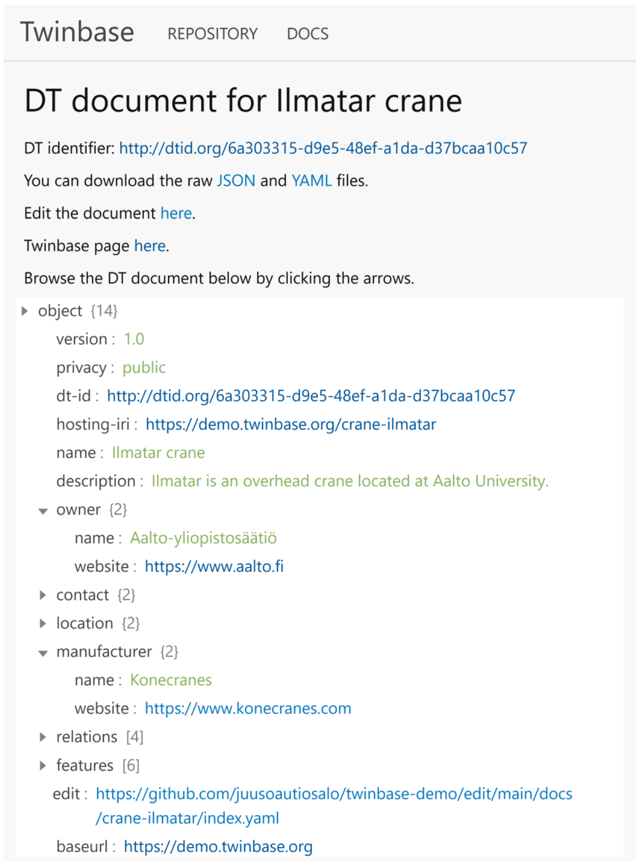
an API to the redirection records and enables the use of custom domains in the free-of-charge service. (In general, paid redirection services are disproportionately high-priced for the added value they provide as their pricing does not take DTID registries into account.) Rebrandly also includes a text note in the redirection record, which is used to store information about the owner of that DTID. Outsiders can add their records manually through a template at https://dtid.org. The main downside of the dtid.org registry is that it relies solely on the authors, which may not seem reliable enough for outside parties, especially from the perspective of the permanence of the identifiers.

B. USER INTERFACES OF TWINBASE

Twinbase supports both human and machine user interfaces, presented in two following subsections.

1) HUMAN USER INTERFACE

Twinbase uses the graphical user interface of a static website as its primary client-side human user interface. The web interface enables browsing the twins as shown in the example front page of Twinbase (Fig. 5) and a selected twin page (Fig. 6). For actions that require creating and modifying DT documents, Twinbase directs users to correct locations in GitHub's web interface. Experienced users can also use any Git client to fetch, modify, and create documents. New Twinbase instance can be created in GitHub according to the instructions in the README.md file. Therefore, Twinbase currently supports four main actions for human users: browsing twins, creating a new Twinbase, creating a new twin, and modifying a twin, which are further specified in Table 1.



Twinbase REPOSITORY DOCS

DT document for Ilmatar crane

DT identifier: http://dtid.org/6a303315-d9e5-48ef-a1da-d37bcaa10c57

You can download the raw JSON and YAML files.

Edit the document here.

Twinbase page here.

Browse the DT document below by clicking the arrows.

```
▸ object {14}
    version : 1.0
    privacy : public
    dt-id : http://dtid.org/6a303315-d9e5-48ef-a1da-d37bcaa10c57
    hosting-iri : https://demo.twinbase.org/crane-ilmatar
    name : Ilmatar crane
    description : Ilmatar is an overhead crane located at Aalto University.
  ▾ owner {2}
      name : Aalto-yliopistosäätiö
      website : https://www.aalto.fi
  ▸ contact {2}
  ▸ location {2}
  ▾ manufacturer {2}
      name : Konecranes
      website : https://www.konecranes.com
  ▸ relations [4]
  ▸ features [6]
    edit : https://github.com/juusoautiosalo/twinbase-demo/edit/main/docs
           /crane-ilmatar/index.yaml
    baseurl : https://demo.twinbase.org
```

FIGURE 6. The twin page of an overhead crane. The DT document of the crane on the white background has been partially expanded by clicking the arrows on the left. The interface was accessible via the DTID http://dtid.org/6a303315-d9e5-48ef-a1da-d37bcaa10c57 at the time of the writing.

2) MACHINE INTERFACE

The static web server of Twinbase provides an HTTP(S) machine interface as it hosts DT documents as YAML and JSON files with standardized names in standard locations. This interface only provides a read interface to the documents, and the machine users need to use a Git client to create and modify the DT documents in the Git repository. There is no machine interface for creating a new Twinbase.

We additionally developed a Python client library that only supports fetching the hosting URL and the DT document with a DTID but can be later developed to include also other operations. The source code for the Python client is available at https://github.com/juusoautiosalo/dtweb-python and the library is also available directly from Python Package Index as "dtweb."

V. PERFORMANCE MEASUREMENTS

To evaluate the basic functionality of DTW, GADIT, and Twinbase, we conducted two types of performance measurements. First, we compared the DT document fetch times of eight DTID registries (Section V-A), and second, we measured the fetch times of a series of six networked twin documents for three registries (Section V-B). Hosting servers

TABLE 1. Four actions supported by Twinbase human and machine interfaces along with additional actions and access requirements. Numbered lists indicate alternative methods in the order of intended convenience.

Action	Human interfaces	Machine interfaces	Additional actions	Access requirements
Browse twins	1. Twinbase website 2. GitHub 3. Git client	1. Python library 2. HTTP request, twin documents available in YAML and JSON formats 3. Git client	-	1. Anyone 2. (Twinbase can be customized to build an access restricted solution.)
Create new Twinbase	GitHub	-	Optional: reserve a domain from a domain registrar	Must have a GitHub account
Create new twin	1. Twinbase website and GitHub 2. GitHub 3. Git client	Git client	Required: create a DTID registry entry for the new twin.	1. Twinbase owner can create new twins 2. Anyone with GitHub account can propose a new twin, the proposal is accepted or rejected by Twinbase owner
Modify a twin	1. Twinbase website and GitHub 2. GitHub 3. Git client	Git client	-	1. Twinbase owner can modify their own twins 2. Anyone with GitHub account can propose a modification, the proposal is accepted or rejected by Twinbase owner

were not compared because Twinbase currently supports only GitHub Pages, and initial tests showed consistent fetch times at approximately 0.1 seconds also for other hosting providers (Domainhotelli and users.aalto.fi were tested). The python scripts used for the measurements are available from GitHub at https://github.com/juusoautiosalo/dtweb-measurements. The violin plot method [69] was used for plotting as it provides a granular view of the measurement data, showing that the samples did not follow a normal distribution. The twins used for the measurements were served from a single Twinbase hosted by GitHub Pages at https://juusoautiosalo.github.io/twinbase-for-measurements.

A. DTID REGISTRY COMPARISON

Figure 7 shows the total time of fetching a DT document with different DTID registries. The time consists of two parts: first fetching the hosting URL of the DT document from a DTID registry, and then fetching the document from the static web server that hosts Twinbase. The fastest registry was d-t.fi with a median of 0.4 seconds, and the slowest was w3id.org with a median of 1.15 seconds. The variation of fetch times inside each registry was more consistent across registries, as the first 99% of samples arrived in a time window of 0.4 to 0.6 seconds for all registries.

B. FETCHING A NETWORK OF TWINS

Fetching a network of twins represents a use case where a user wants to fetch DT documents that are consecutively referenced in the DT documents, such as searching for keywords across the contents of related twins. Figure 8 shows the measurement results for the fetch times of an origin DT and its five consecutive children. Three registries were selected for network measurements: d-t.fi was among the fastest ones, whereas dtid.org and w3id.org have unique functionalities

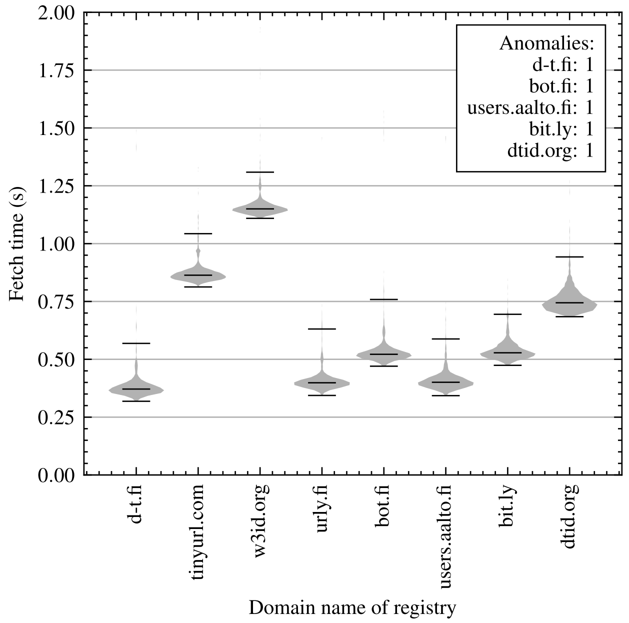


FIGURE 7. Total times for fetching DT documents with DTIDs at different DTID registries. The times include both fetching the hosting URL and the DT document. The gray areas represent the distribution of samples with the violin plot method, and the black horizontal lines from lower to higher are quantiles 0.00 (minimum), 0.50 (median), and 0.99. The sample size was 1000 for each registry, constituting a total of 8 000 fetches for the measurement. Samples over 2 seconds were excluded from the plot and reported as anomalies. All documents were hosted at the same Twinbase hosted by GitHub Pages, giving consistent fetch times of about 0.1 seconds.

and represent the slow end. The measurement script implemented the measurements in the following phases:

0) Fetch DT document of origin DT (depth 0) and read the DTID of its child DT (depth 1).

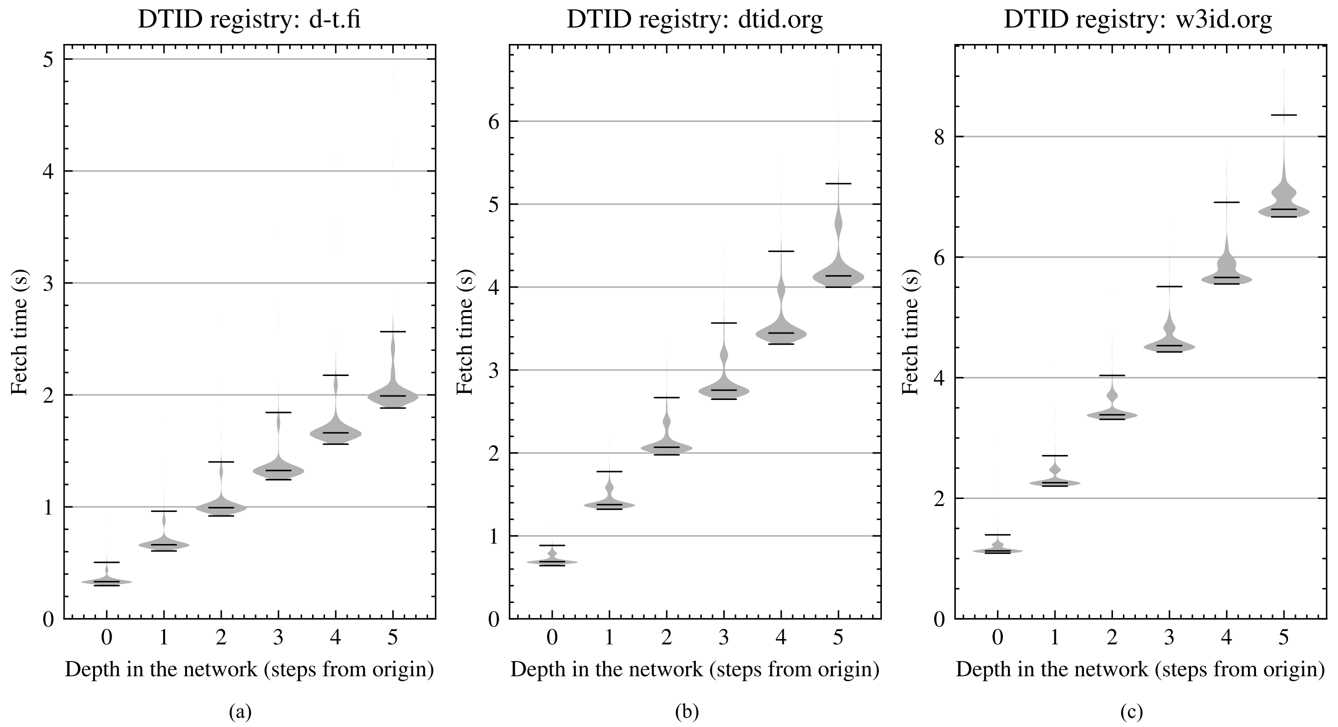


FIGURE 8. Total times for fetching DT documents referenced in other DT documents. The gray areas represent the distribution of samples with the violin plot method, and the black horizontal lines from lower to higher are quantiles 0.00 (minimum), 0.50 (median), and 0.99. The measurement started by fetching the DT document of the origin DT (depth 0) and continued to its child (depth 1) and so on until the last child (depth 5) had been fetched. The sample size is 1000 for each DT in each registry, constituting a total of 18 000 fetches for the three measurements. All documents were hosted at the same Twinbase hosted by GitHub.

1) Fetch DT document of child DT (depth 1) and read the DTID of its child DT (depth 2).
2-4) Fetch DT document of child DT (depth n) and read the DTID of its child DT (depth n + 1)
5) Fetch DT document of origin DT (depth 5) and notice that it has no children.

The last DT document arrived at the client at a median of about 2.0 seconds for d-t.fi, 4.1 seconds for dtid.org, and 6.8 seconds for w3id.org. The arrival times grew linearly for each registry.

VI. DISCUSSION

Twinbase and other DTW solutions presented in this article appear to be capable tools for building a global network of digital twins but can be improved in several ways and should be validated in real-life use cases. The following subsections discuss the presented solutions from the viewpoints of performance, usability, security, standardization, openness, and as a tool to build the Digital Twin Web.

A. PERFORMANCE

The performance of the static web servers used for Twinbase was fast and consistent with response times of approximately 0.1 seconds, but DTID registries introduced significant latencies to the total DT document fetch time as shown in Fig. 7. For the three fastest registries, approximately half of the documents arrived in less than approximately 0.4 seconds, and 99% of the samples arrived in less than 0.65 seconds, whereas for the slowest registry, the same quantiles took place at 1.15 and 1.30 seconds. As the sample size was 1000, this can be considered a reliable result with the test setup. The results were also consistent between various measurement runs conducted on different days. As such, this difference clearly impacts the user experience of browsing the DT documents based on the DTIDs.

The difference multiplies when fetching networked twins as the median for the fetch time of the depth 5 DT document is approximately 2.0 seconds for the fastest registry and 6.8 seconds for the slowest. Fetching six levels of DTs is decent with the fastest registry but completely impractical with the slowest. This type of network fetch might be performed, e.g., when searching for a temperature sensor from a room whose devices have interlinked DT documents. The search process might include iterating with the search terms similarly as people conduct Internet searches. The iterative searches would be frustratingly slow with the fastest registry and unmanageable with the slowest.

The performance measurements demonstrate that the presented DTID registry approach is sufficient for some types of basic applications but clearly too slow for more complex use such as search. Therefore, improvements will be welcome, especially on the response times of DTID registries. Incremental improvements can enhance performance up to a certain point, but even the whole approach could be rethought. One approach would be to build a system similar to the Domain Name System for DTIDs. Even the existing DNS infrastructure could be used if the identifying part of DTIDs would be put into a subdomain, although this would require new software for the DTID registry. Another approach is to build search indexes of DTW similarly as search engine providers, such as Google, are building search indexes of WWW resources. The users of a DTW index would search from the index through its interfaces and access the content via the actual DTW servers. Building a search index is possible with the current DTID and Twinbase solutions, although a clear flag that a page is part of DTW would help make the index with crawlers. Also, an intelligent cache service akin to the technologies used by Google and product recommendation systems could help improve the user experience of the iterative searches. As related work, a discovery service has been proposed for the Web of Things Thing Description [70] and a Google-style search service called Swoogle has been made for the Semantic Web [71].

Outside the conducted measurements, we assume that the number of records in a DTID registry should not affect their performance. The performance of updating DTID records was not measured either but seemed responsive enough to be unnoticed by human users. Measuring the baseline performance of both of these metrics would be useful for planning future work. However, these performance metrics depend on the implementation of each DTID registry, and we recommend developing the fundamental technologies of the registries further before starting to optimize their performance.

B. USABILITY CONSIDERATIONS

In earlier work, ease-of-use was found as one of the most important characteristics of tools that are used for building digital twins [56]. Twinbase leverages this observation by providing an as easy-to-use platform for distributing DT documents as possible. However, Twinbase is only at an early stage of development and its usability can be improved in various ways, as described in the following subsections.

1) BROWSING THE TWINS

Currently, Twinbase provides a web interface to browse the twins, which means that the documents can be accessed with virtually any internet-capable device, including any phone and laptop, without installing new software. The web interface shows some basic information about each twin and presents the DT document in collapsible tree form. This interface provides a basic level of usability even for people who are not experts in programming. However, no particular thought has been put on how the information on the twin page should be arranged to provide the most useful information as efficiently as possible. For example, users might find it convenient if the most used features and links were brought to the top of the twin page, even if they happen to be at the bottom in the DT document. Furthermore, the relations between the twins could be presented as a graphical web of twins, similarly as with the Azure Digital Twins Explorer. As even further development, it would be interesting to test the usability of a mixed reality explorer app that presents the contents of a DT document when it detects a QR code that contains a DTID.

A DT document can provide a lot of information about the twin and its real-world counterpart. However, this can lead to an information overload for the average users of digital twins. For example, when a restaurant visitor scans a DTID at the door, they probably don't care about most details expressed in the DT document. Rather, they might like to be directed directly to a page that displays the menu of the restaurant. A link to the Twinbase page could be added to that page so that thorough customers can check any details that interest them. This approach could also be adjacent to an external DT reader app that crawls the DT document according to user preferences, for example, letting the visitor know right away if the restaurant has vegan-friendly meals or uses meat that has been certified according to animal welfare standards. Average CO2 emissions and calories per meal could also be available. These approaches require further development of the techniques expressed in this paper as well as domain-specific standardization but are still very much possible with Twinbase as the root server for DT documents.

2) EDITING THE TWINS

Twinbase adds a link to the GitHub editing interface to each DT document, so users need only one click from the twin page to start editing the document. Nevertheless, users still need to know what to write on the document. To provide better guidance, Twinbase could provide an editor that suggests adding popular fields to the twin and makes sure that the document is syntactically correct before pushing it to the repository. The workload for implementing this kind of embedded editor should be manageable by creating a JSON Schema [72] for the DT document and leveraging that in conjunction with the Ace editor [73] like JSON Editor Online [74] does, but for YAML files similarly to the YAML extension for Visual Studio Code [75]. JSON Schema is an open specification, and the other tools are permissibly licensed open-source software, so they can be modified if necessary.

On the machine user side, there should be a library that enables editing DT documents stored in Twinbase. Furthermore, the time it takes from making an edit until the edit is visible on the twin page should be made faster and more consistent.

3) CREATING NEW TWINS

Twinbase provides a guided process to create a new DT document through a link on the front page. However, adding fields to the document is not a guided process. If the creator wants any informative fields in addition to the basic fields added by the new twin creation process, the creator needs to browse the DT document specification and manually add the necessary fields. As an improvement to this approach, it would be useful to have DT document templates for various types of twins and even have a DT document initializer that allows the creator to add appropriate fields from a list. As a technical improvement to the new twin creation process, the latency from creating the twin to it being accessible on the server should be lowered.

4) CREATING NEW TWINBASE

A public Twinbase instance can be created without configuring any server hardware by using free-of-charge GitHub services. A new instance of Twinbase is created by forking the repository or using the repository as a template, and the newly created repository will launch itself up after the new owner has enabled GitHub Actions and Pages. The Twinbase template repository includes instructions on how to create a new Twinbase.

The Twinbase creation process could be made easier through automation, but in the opinion of the authors, it is already easy enough as creating a new Twinbase is fairly rare. Instead, an update mechanism should be put in place as soon as possible.

C. SECURITY CONSIDERATIONS

The security of the solutions proposed in this article is at a decent level, but there are many opportunities for improvement. The solutions rely on two primary sources of security: traditional web security and the security measures taken by the Git provider. The traditional web security applies mainly to the user interface, whereas the Git provider takes care of back-end security. We also present methods on how to improve the security of DTIDs with cryptographic measures. The solutions also potentially open up new attack vectors from a privacy perspective, but their investigation is left for further studies.

The traditional web security measures of GADIT servers include the HTTPS protocol and the Domain Name System (DNS), although there are many more security mechanisms in various layers of the Internet. HTTPS is an extension of the HTTP protocol that relies on certificate authorities to ensure the privacy and integrity of the network traffic between the DTW server and the client. HTTPS is not the most foolproof mechanism of ensuring security as it relies on separate authorities, but it does provide a significant upgrade from normal HTTP. DNS resolves the domain name of a URL into an IP address. DNS is used by DTW for two tasks: the current DTIDs rely on DNS as a root authority, and the routing of traffic to the hosting servers is handled by DNS. There are a number of possible attacks on the DNS system, such as DNS spoofing, hijacking, and rebinding. These problems mainly stem from the time when the Internet was considered a safe space, and now there are ways to prevent these attacks, for example, DNSSEC (Domain Name System Security Extensions) and DNS over HTTPS. DNSSEC verifies the integrity of DNS records via cryptographic authentication, and DNS over HTTPS increases privacy. In conclusion, traditional web security methods provide the same basic level of security as the web that we so commonly use, but as digital twins are expected to be used for critical physical world functions, DTW technologies and standards will need to go through an additional risk analysis.

Git providers act as centralized authorities that ensure the security of Git repositories in contrast to the distributed security measures of the web. It is clear who is responsible for the security of a Git repository, which has likely contributed to the existence of good security practices in Git services. As baseline security, Git providers require sign-in to make modifications to the Git repository, but on top of that, e.g., GitHub allows users to enforce two-factor authentication for their accounts. Git repositories may be owned by a single user or by an organization that has multiple members. In GitHub, an organization may require two-factor authentication for all of its members. As a recent feature, GitHub has enabled users to verify their commits (i.e., edits to Git repository) with a signature, so viewers can have additional verification on who created the commit. Overall, the security level of Git services can be considered good, and they can and have been trusted for critical functions. Alternatively to using external Git providers, it is also possible to self-host a Git repository, although this approach has imposed serious security vulnerabilities [76].

In addition to the existing security practices of DTW, GADIT, and Twinbase, there are multiple opportunities for improvement. As an apparent addition, Twinbase should enable the creation of a private server. Luckily, there seem to be no obstacles for creating private servers. The authors decided to start with public server implementation mainly to enhance the adoption rate of DTW through visibility, and while the public servers also have the advantage of leveraging free-of-charge services, creating a private Twinbase that works on private hardware should be just a matter of work.

In the future, cryptographic measures could be used to ensure the integrity of DT documents through signature procedures. For example, the DT identifier could include the hash of a public key, and the private key could be used to sign the DT document. As the public keys and their hashes should (for security reasons) be long enough to be unique, DTID could be defined as `[domain]/[(hash of) public key]`. DTW server owner would then sign all DT documents, and DT document users can check that the document has been signed by the issuer of the DTID. This works as long as the holder of the private key can be trusted by the DTW server owner. However, this is not the case if the DTID is generated by the owner of a twin and the twin is sold to a new owner. Hence, the private key should be held by a neutral, trusted party, such as a DTID registry operator. In this case, the registry operator would sign the DT documents at the request of the DTID owner. This style also poses a risk that a DTID becomes permanently insecure if the private key is compromised.

As an alternative to permanently attaching the public key to the DTID, the public key could be added as an extension. The WWW standards already have defined *query strings* [77] as a method for sending parameters as parts of URL. Hence, the DTID coupled with an extension would be in the form: `[domain]/[unique identifier]?[algorithm id]=[(hash of) public key]`. A query string is the part of the URL after the ? sign, and it consists of a key-value pair divided by = sign. The algorithm id can be used to define the version of the signing algorithm used. It should be standardized so that both the client and the server know what algorithm to use when ensuring the integrity of the DT document. The query string should be included with the DTID whenever the user might want to ensure the integrity of the DT document behind the DTID. For example, it may be convenient to include the string in a QR code that is attached to the physical counterpart of the twin. A query string does not create a conflict with redirection, so it can be used truly as an extension of the DTID, working even with DTID registries that might not specifically support this feature. However, it may cause confusion about when the key should be used and who should hold the private key.

As a further development goal, digital twins could use self-sovereign identities to be independent of identifier issuing authorities. These methods are being developed by the authors. Also, the use of role-based access control (RBAC) and attribute-based access control (ABAC) methods should be investigated for securely providing access either to complete records or to specific fields that should not be publicly visible.

D. STANDARDIZATION CONSIDERATIONS

This article has discussed two topics that are considered subjects of standardization: the Digital Twin Web in general and the DT document. The DTW needs to be standardized so that users all over the web know how to find and access DT documents, while the contents of DT documents need to be standardized so that the users can leverage them efficiently. The two following subsections do not define the standards but rather describe what should be standardized.

1) DTW STANDARDIZATION

The Digital Twin Web is at its inauguration, and the topics for standardization should start from the very basics. There are three main topics: DTID, DT document, and the protocol on how to fetch a DT document based on its DTID. The topics are similar to the WWW, which has the domain name system, HTML documents, and HTTP protocol. Each of the three DTW standardization topics contains several details that need to be standardized. The standardization process can be similar to the WWW as well, i.e., an iterative process involving both technical implementations and standardization efforts. Despite the similarities, there are two main differences between DTW and WWW. First, the WWW was built on top of early Internet standards, whereas DTW is mainly built on top of WWW. Second, while the WWW was made to contain pretty much any digital resources, the DTW is made only for digital twins. These differences pose both advantages and difficulties for DTW development. It is useful to have the WWW as a basis, but some of its solutions may not be optimal for DTW, which means some parts of the DTW standards need to be very detailed. A digital twin is also more conceptually difficult to grasp than a web page. While having a "twin page" (Fig. 6) may provide enough value for some applications, for many, it might be too little.

We expect the DTW standards to become a collection of multiple standards rather than a single document. One development direction is to become a collection of a very large amount of small definitions similar to the Request For Comments (RFC) documents upkept by the Internet Engineering Task Force. An RFC is not normative by default, but some of the RFCs are promoted to the level of Internet Standard if they are mature enough and provide significant benefit to the Internet community. This type of approach seems appropriate also for the DTW, although the community should be digital twin specific.

2) DT DOCUMENT STANDARDIZATION

Defining a digital twin-specific information model is one of the most important DTW standardization activities, enabling the efficient use of DT documents across the diverse applications of the web. Standardization must include both the default format of the document and the contents that describe the actual information model.

The format needs to provide both technical functionality and user-friendliness. Twinbase currently uses YAML for DT documents thanks to its user-friendliness, even in raw text form. However, plain YAML is not technically capable of describing ontologies, so it should be extended to include similar properties as JSON-LD. While waiting for a standardized linked data format in YAML, we recommend that JSON-LD be used as the master format of the DT document, and the YAML file can be used as an interface by converting JSON-LD to YAML and back.

The digital twin information model standard will likely be a mix of digital twin-specific and domain-specific ontologies. This work concentrates only on the DT-specific ontologies, i.e., defining the default variables of all DT documents. A basic goal is that a DT document always includes the information that meta-level digital twins need to function properly. The necessary information can mean both universally mandatory terms and terms that are mandatory only on specific platforms, such as Twinbase.

DTID, along with the whole DTID registry record, proved to be an integral part of the DTW. While the master data lies in the DTID registry, we recommend that each DT document contains a copy of its DTID record:

- DTID (permanent),
- hosting IRI (changeable), and
- the owner of the DT (preferably the DTID of the owner, changeable).

When the record is stored in both places, the DTW server can periodically check the validity of the record, which might be especially important to avoid data loss in the beginning as DTID registries are still being developed.

Many further topics need to be standardized before DT documents can be used efficiently, and we are able to mention only some of them as our work concentrated on platform development instead of application development. The method(s) to define relations to other twins should be standardized promptly to be able to create proper networks of digital twins. The coupling to the real-world counterpart is generally considered as the main requirement for all digital twins, and hence, further work should develop a set of methods to identify the real counterpart of the twin and include them as standard entries in the DT document. Contact information is another useful term that may affect the functionality of the DTW in general. As further development, it might be useful to define different mandatory fields for different types of DTs. For example, a person might have different mandatory fields than an industrial machine.

E. OPENNESS CONSIDERATIONS

In the authors' opinion, building a network of twins on a scale similar to the WWW will only be possible if DTW standards are similarly free and openly accessible as the standards of the WWW. More specifically, the open standards of DTW need to describe the communication protocols in such detail that a new DTW server or client can be developed from scratch. If these standards were in place, Twinbase would not need to be open source, but at the current stage, it is very useful to have an open-source DTW platform to show where the needs for the standards come from. Furthermore, having an open-source server implementation for a DTW server will likely promote its growth and adoption.

Open source can also aid in synchronizing the lifetimes of digital and physical products. Digital products have a notoriously short life span which does not match the long lifespan of physical products. For example, an industrial crane might be fulfilling its purpose just fine after fifty years of service, whereas a web browser has reached the end of support long before the five-year mark. Digital products can be updated relatively easily, but this does not help if the provider decides to discontinue their closed-source service or the updated mainline software becomes incompatible with certain physical products. With open-source software products, users can buy updates from an alternative provider or update the software themselves. For example, if the authors would discontinue Twinbase development for some reason, a community of users could start updating it, and the physical products leveraging Twinbase would stay functional.

F. BUILDING THE DIGITAL TWIN WEB WITH TWINBASE

Digital Twin Web is supposed to become a primarily open network of digital twins, i.e., the fabric connecting the physical and digital worlds together. Achieving this kind of prominent open network requires that a critical mass of users receive value from the time and money they invest in DTW solutions. Twinbase aids in gathering that mass by lowering the total cost of publishing DT documents. The value proposition of DTW is not yet crystallized and will inevitably vary by use case, although most cases will probably include optimizing the use of physical resources by providing the right information at the right place at the right time. The cases need to be developed one-by-one until we have libraries of standard solutions.

At the current stage, extended use of Twinbase is hindered by the lack of comprehensively defined DTW standards. The most immediate need seems to be a DT document standard that has provided value in real-world applications. The standards can only be specified after gaining experience of what works and what does not, and we expect Twinbase to be a useful tool for developing the standards because its user-friendliness allows a high number of iterations by a large group of users.

Twinbase also points out concrete technical requirements for the DTW standards as it is a working technological solution instead of a conceptual architecture. In addition to identifying needs for technical details such as having a protocol to transfer DT documents, the overall architecture of Twinbase has shown the demand for various external services. DTID registry is considered mandatory in the current architecture, but there can also be others, such as a search index that enables the efficient discovery of DT documents. The experimentation with various services starts forming the larger picture of how the first value-adding version of DTW will actually work.

Twinbase makes the need for publicly available twins apparent. Each house, street, city, country, and continent should have its own twin so that other twins can describe their relationship with them. This ability enables positioning the twins to cyberspace in a similar manner that real-world entities are positioned in regard to each other, leading to a situation where the DTW starts to resemble the real world. Public twins may contain only little information, as the mere existence of a digital twin of a street is enough to link the houses on that street together in the DTW. Privacy is also ensured as owners of twins can decide the publicity level of their twins, so all objects inside a house can be entirely private. Nevertheless, the new ability to describe real-world things on the Internet brings up a question of where the boundaries between public and private information should stand. The conversation has already started with the images saved in Google Street View, and the development of DTW will likely hasten this discussion. Nevertheless, we hope that public meta-level digital twins provide a shared digital infrastructure on which other solutions can be built.

Ownership of digital twins deserves special attention. We suggest that each twin is owned by the party that owns the real-world counterpart. For example, the digital twin of a privately owned thing, such as a crane, should be owned by the owner of the physical crane, and the manufacturer can describe extensions to that twin in its own systems via the permanent DTID. An extension is a twin as well and owned by whoever creates the extension, such as the manufacturer or any other user who wants to attach information to the crane. Extensions can be private or public. Public goods should have their twins too, and the corresponding public institutions should own them. For example, the twins of the public streets located in the City of Espoo should be owned by the City of Espoo. However, Twinbase enables the creation of public digital twins by anyone for anything, and hence ''rogue'' twins may start appearing if the real owners are not creating the necessary twins. Then it is the responsibility of users to determine whose digital twin to use and trust, and in the lack of better options, some rogue twins may become the ''de facto'' public twins. It will probably take years before this problem becomes timely for digital twins, but on the other hand, citizens already receive a large part of their information on public infrastructure from private solutions such as Google Maps.

VII. RISKS AND LIMITATIONS

This article has shown that DTW and Twinbase can work technically in limited use but require further development before providing value to users. To reach full potential, the development of DTW will require a wide community of users, which might be difficult to attract as there are no concrete examples of value-adding use cases. However, related work demonstrates interest in developing meta-level digital twins from many fronts, so the conceptual and technical experimentation presented in this article should provide useful general knowledge even if the Twinbase platform itself is not used.

The DT document standard is at the draft stage and heavily under development. The structure of the document is unfinished, and some aspects of the standard will likely be backward incompatible. Hence, the documents along with Twinbase cannot yet be used to their full potential, and some of the work put into creating them will likely have to be redone. The DT document standard is planned to be merged or cross-used with other specifications, but technical or other difficulties might arise to prevent this.

Currently, Twinbase does not allow hosting DT documents privately, which means that it cannot be used for sensitive information, leaving out many potential use cases. However, public-first is a conscious strategical choice that should attract a critical mass of public DT documents that act as infrastructure for the private ones. Creating private implementations of Twinbase will require some development but should be possible.

The hosting URL can be mistakenly used for referring to the DT instead of using the DTID. The URL works at first but will cause problems when the hosting URL changes due to, e.g., DT ownership change, transfer to another Twinbase, or other change in base hosting URL. This problem could be prevented by providing proper client libraries and clear instructions to users or by developing a DTID registry that allows twin documents to be hosted from the DTID.

VIII. MANAGERIAL IMPLICATIONS

Should your company start using Twinbase right now? In production, no. In the research department, we recommend getting familiar with DT document standards and their distribution technologies, such as Twinbase. During your exploration, it is good to keep in mind that the benefits of DT documents vary drastically by use case, and the best business cases may arise from anywhere in the company, e.g., from sales.

While Twinbase and the whole Digital Twin Web initiative are in the very early days of their development, the race to define many related standards is already going strong as big industry players are describing their digital twins in various formats, including AAS, DTDL, and WoT TD. These formats describe knowledge on various layers and have their own emphasis points, so the competition between them may not become a VHS vs. Betamax style either-or competition, but rather each standard may find its place in the future set of digital twin standards.

Current commercial digital twin platforms seem to be missing a method to create permanent, globally accessible, and transferable identifiers for digital twins. These identifiers are not needed for digital twin networks that are limited to one company or consortium, so within these boundaries, it is possible to solve complex technical tasks with digital twin networks created with the existing platforms. However, to reach a Digital Twin Web that opens up a similar multitude of business opportunities as the World Wide Web has opened in the last thirty years, a global standard for proper digital twin identifiers must be developed. The DTID registry concept presented in this paper is one method for creating good enough identifiers, but future developments may show systems similar to the DNS or even new types of identifiers conforming with the Self-Sovereign Identity concept.

Defining the publicity level of the DT documents needs special attention. Publicly available DT documents might attract developer communities across the globe and turn your products into thriving business platforms, but to remain competitive, companies need to categorize the publicity level of their meta-level digital twins strategically. While business-critical information naturally stays internal or with trusted partners, not all remaining material should be published because public data needs to be updated carefully, increasing maintenance costs. Companies need to develop processes on how to add new material to public DT documents to avoid leaking confidential information and to identify metadata that leads to increased cash flow when published.

IX. CONCLUSION

This article introduced Twinbase, its underlying architecture GADIT, the basic structure of Digital Twin Web, and the concept of digital twin identifier (DTID) registry. These solutions can be used to distribute digital twin documents from owners to users, as displayed in Fig. 1. We implemented initial performance measurements and conclude that the overall approach seems promising but should be improved in several ways described throughout the article. The solutions also need to be validated and guided by real-life use cases. To promote the adoption of Digital Twin Web technologies, we introduced four openly accessible resources, listed in Table 2.

TABLE 2. Openly accessible resources introduced in this article.

Resource	Available at
Twinbase source code	https://github.com/twinbase/twinbase
Twinbase demo server	https://dtw.twinbase.org
DTWeb python library	"dtweb" via pip or https://github.com/juusoautiosalo/dtweb-python
Python scripts for performance measurements	https://github.com/juusoautiosalo/dtweb-measurements

Twinbase is an open-source platform for distributing digital twin documents as effortlessly as possible. People with no experience in programming or server administration can create a public and free-of-charge instance of Twinbase using only a web browser. Twinbase leverages GitHub repository, GitHub Actions, and GitHub Pages by default but can be modified to use their alternatives.

GADIT is a general architecture that describes how a Git repository can be used as a Digital Twin Web server when coupled with a CI system that updates files and a static web server where the contents are deployed. GADIT was used for Twinbase but can also be used to create alternative implementations.

Digital Twin Web is a proposed information system where digital twins are described by DT documents, identified by digital twin identifiers, and accessible over the Internet. The structure of the Digital Twin Web highly resembles the structure of the World Wide Web, but in distinction, each twin mirrors a real-world counterpart. The twins should be interlinked according to the relations of their real-world counterparts so that the accumulation of the twins starts to mirror the real world. Digital Twin Web is formed through a combination of standards and technologies whose development will be a demanding task.

DTID registries store permanent, globally accessible, and transferable identifiers for digital twins. The concept requires further development, and as proper DTID registries do not exist yet, we used a selection of URL redirection services in their place.

Performance measurements suggest that the overall approach is feasible for simple use cases, but more complex cases demand further development. The static web servers used for Twinbase proved to be consistently fast with a response time of approximately 0.1 seconds across three server providers, but the URL redirection services used as DTID registries showed surprisingly great and consistent variation. The median times for fetching a DT document ranged from 0.4 seconds for the fastest registries to 1.1 seconds for the slowest. When fetching a series of six interrelated twins, the median time for the last document to arrive was 2.0 seconds for the fastest registry and 6.8 for the slowest.

Twinbase is still work-in-progress, and its usefulness is limited by the capabilities of the DT document standard. However, by providing an effortless solution for hosting public DT documents, Twinbase may be able to boost DT document standard development. A prominent next step would be to map existing standards [27] to the DT document standard draft and to automatically convert between them on the Twinbase server.

ACKNOWLEDGMENT

The authors would like to thank Pekka Nikander for bouncing ideas on how to enhance security with cryptography. JUUSO AUTIOSALO also would like to thank KAUTE Foundation and Walter Ahlström Foundation.

REFERENCES

[1] 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, doi: 10.1007/978-3-319-38756-7_4.

[2] M. Shafto, M. Conroy, R. Doyle, E. Glaessgen, C. Kemp, J. LeMoigne, and L. Wang, "Modeling, simulation, information technology & processing roadmap," NASA, Washington, DC, USA, Tech. Rep. 11, 2010. [Online]. Available: http://www.nasa.gov/pdf/501321main_TA11-MSITP-DRAFT-Nov2010-A1.pdf

[3] E. Glaessgen and D. Stargel, "The digital twin paradigm for future NASA and US Air Force vehicles," in *Proc. 53rd AIAA/ASME/ASCE/AHS/ASC Struct., Struct., Dyn. Mater. Conf.*, Honolulu, HI, USA, Apr. 2012, p. 1818, doi: 10.2514/6.2012-1818.

[4] F. Tao and M. Zhang, "Digital twin shop-floor: A new shop-floor paradigm towards smart manufacturing," *IEEE Access*, vol. 5, pp. 20418–20427, 2017, doi: 10.1109/ACCESS.2017.2756069.

[5] F. Tao, F. Sui, A. Liu, Q. Qi, M. Zhang, B. Song, Z. Guo, S. C.-Y. Lu, and A. Nee, "Digital twin-driven product design framework," *Int. J. Prod. Res.*, vol. 57, no. 12, pp. 3935–3953, 2019, doi: 10.1080/00207543.2018.1443229.

[6] M. Langheinrich, F. Mattern, K. Römer, and H. Vogt, "First steps towards an event-based infrastructure for smart things," in *Proc. Ubiquitous Comput. Workshop*, 2000, p. 34. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.10.9512

[7] M. Kärkkäinen, T. Ala-Risku, and K. Främling, "The product centric approach: A solution to supply network information management problems," *Comput. Ind.*, vol. 52, no. 2, pp. 147–159, Oct. 2003, doi: 10.1016/S0166-3615(03)00086-1.

[8] D. McFarlane, S. Sarma, J. L. Chirn, C. Y. Wong, and K. Ashton, "Auto ID systems and intelligent manufacturing control," *Eng. Appl. Artif. Intell.*, vol. 16, no. 4, pp. 365–376, Jun. 2003, doi: 10.1016/S0952-1976(03)00077-0.

[9] J. C. Camposano, K. Smolander, and T. Ruippo, "Seven metaphors to understand digital twins of built assets," *IEEE Access*, vol. 9, pp. 27167–27181, 2021, doi: 10.1109/ACCESS.2021.3058009.

[10] S. H. Khajavi, N. H. Motlagh, A. Jaribion, L. C. Werner, and J. Holmstrom, "Digital twin: Vision, benefits, boundaries, and creation for buildings," *IEEE Access*, vol. 7, pp. 147406–147419, 2019, doi: 10.1109/ACCESS.2019.2946515.

[11] Q. Lu, A. K. Parlikad, P. Woodall, G. Don Ranasinghe, X. Xie, Z. Liang, E. Konstantinou, J. Heaton, and J. Schooling, "Developing a digital twin at building and city levels: Case study of west Cambridge campus," *J. Manage. Eng.*, vol. 36, no. 3, May 2020, Art. no. 05020004, doi: 10.1061/(ASCE)ME.1943-5479.0000763.

[12] B. R. Barricelli, E. Casiraghi, J. Gliozzo, A. Petrini, and S. Valtolina, "Human digital twin for fitness management," *IEEE Access*, vol. 8, pp. 26637–26664, 2020, doi: 10.1109/ACCESS.2020.2971576.

[13] R. Saracco, J. Autiosalo, D. de Kerckhove, F. Flammini, and L. Nisiotis. (Apr. 2020). *Personal Digital Twins and their Role in Epidemics Control.* [Online]. Available: https://digitalreality.ieee.org/images/files/pdf/PDT-role-in-Epidemics_FINAL.pdf

[14] G. Pappas, J. Siegel, and K. Politopoulos, "VirtualCar: Virtual mirroring of IoT-enabled avacars in AR, VR and desktop applications," in *Proc. Int. Conf. Artif. Reality Telexistence Eurograph. Symp. Virtual Environ. Posters Demos*, T. Huang, M. Otsuki, M. Servières, A. Dey, Y. Sugiura, D. Banakou, and D. Michael-Grigoriou, Eds. Stockholm, Sweden: The Eurographics Association, 2018, pp. 1–13, doi: 10.2312/egve.20181381.

[15] J. E. Siegel, "Cloudthink and the Avacar: Embedded design to create virtual vehicles for cloud-based informatics, telematics, and infotainment," M.S. thesis, Dept. Mech. Eng., Massachusetts Inst. Technol., Cambridge, MA, USA, 2013.[Online]. Available: http://hdl.handle.net/1721.1/92230

[16] A. N. Pedersen, M. Borup, A. Brink-Kjár, L. E. Christiansen, and P. S. Mikkelsen, "Living and prototyping digital twins for urban water systems: Towards multi-purpose value creation using models and sensors," *Water*, vol. 13, no. 5, p. 592, Jan. 2021, doi: 10.3390/w13050592.

[17] A. M. Karadeniz, I. Arif, A. Kanak, and S. Ergun, "Digital twin of eGastronomic things: A case study for ice cream machines," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2019, pp. 1–4, doi: 10.1109/ISCAS.2019.8702679.

[18] R. Parmar, A. Leiponen, and L. D. W. Thomas, "Building an organizational digital twin," *Bus. Horizons*, vol. 63, no. 6, pp. 725–736, Nov. 2020, doi: 10.1016/j.bushor.2020.08.001.

[19] 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, pp. 20325–20336, 2019, doi: 10.1109/ACCESS.2019.2897018.

[20] J. Siegel and S. Sarma, "A cognitive protection system for the Internet of Things," *IEEE Secur. Privacy*, vol. 17, no. 3, pp. 40–48, May 2019, doi: 10.1109/msec.2018.2884860.

[21] J. E. Siegel, S. Kumar, and S. E. Sarma, "The future Internet of Things: Secure, efficient, and model-based," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2386–2398, Aug. 2018, doi: 10.1109/JIOT.2017.2755620.

[22] A. Canedo, "Industrial IoT lifecycle via digital twins," in *Proc. Int. Conf. Hardw./Softw. Codesign Syst. Synth.*, Oct. 2016, p. 1. [Online]. Available: https://ieeexplore.ieee.org/document/7750990

[23] S. P. A. Datta, "Emergence of digital Twins—Is this the March of reason?" *J. Innov. Manage.*, vol. 5, no. 3, pp. 14–33, Nov. 2017, doi: 10.24840/2183-0606_005.003_0003.

[24] R. Rosen, J. Fischer, and S. Boschert, "Next generation digital twin: An ecosystem for mechatronic systems?" *IFAC-Papers Line*, vol. 52, no. 15, pp. 265–270, 2019, doi: 10.1016/j.ifacol.2019.11.685.

[25] 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.

[26] R. Ala-Laurinaho, J. Autiosalo, A. Nikander, J. Mattila, and K. Tammi, "Data link for the creation of digital twins," *IEEE Access*, vol. 8, pp. 228675–228684, 2020, doi: 10.1109/ACCESS.2020.3045856.

[27] M. Jacoby and T. Usländer, "Digital twin and Internet of Things-current standards landscape," *Appl. Sci.*, vol. 10, no. 18, p. 6519, Jan. 2020, doi: 10.3390/app10186519.

[28] S. Kaebisch, T. Kamiya, M. McCool, V. Charpenay, and M. Kovatsch. (Jun. 2020). *Web of Things (WoT) Thing Description*. [Online]. Available: https://web.archive.org/web/20210325091547/

[29] Contributors. (Jul. 2020). *Digital Twin Definition Language*. [Online]. Available: https://github.com/Azure/opendigitaltwins-dtdl

[30] Plattform Industrie 4.0. (Nov. 2020). *Details of the Asset Administration Shell*. [Online]. Available: https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/Details_of_the_Asset_Administration_Shell_Part1_V3.pdf?blob=publicationFile&v=5

[31] ETSI. (Feb. 2021). *ETSI GS CIM 009 V1.4.1 Context Information Management (CIM): NGSI-LD API*. [Online]. Available: https://portal.etsi.org/webapp/workprogram/Report_WorkItem.asp?WKI_ID=61461

[32] Harbor Research. (Mar. 2021). *An Open-Source Fabric for Digital Twins*. [Online]. Available: https://harborresearch.com/connected-complexity/

[33] E. Ditto. (Mar. 2021). *Protocol Specification Eclipse Ditto a Digital Twin Framework*. [Online]. Available: https://www.eclipse.org/ditto/protocol-specification.html

[34] World Wide Web Consortium. (2021). *Standards—W3C*. [Online]. Available: https://www.w3.org/standards/

[35] *Git Community*. Accessed: Jul. 26, 2021. [Online]. Available: https://git-scm.com/

[36] GitHub. (2021). *GitHub Features: The Right Tools for the Job*. [Online]. Available: https://github.com/features

[37] A. Bolton, L. Butler, I. Dabson, M. Enzer, M. Evans, T. Fenemore, F. Harradence, E. Keaney, A. Kemp, A. Luck, N. Pawsey, S. Saville, J. Schooling, M. Sharp, T. Smith, J. Tennison, J. Whyte, A. Wilson, and C. Makri, "Gemini principles," Apollo-Univ. Cambridge Repository, Cambridge, U.K., Tech. Rep. CDBB_REP_006, 2018, doi: 10.17863/CAM.32260.

[38] S. Chorlton, T. Hughes, A. Robasto, and T. Au, "GBDT hub annual benchmark report 2020," Center Digit. Built Britain, Cambridge, U.K., Tech. Rep., Mar. 2021. [Online]. Available: https://digitaltwinhub.co.uk/files/file/72-annual-benchmark-report/

[39] E. Barnstedt, B. Boss, E. Clauer, D. Isaacs, S.-W. Lin, S. Malakuti, P. van Schalkwyk, and T. W. Martins. (Mar. 2021). *Open Source Drives Digital Twin Adoption*. [Online]. Available: https://www.iiconsortium.org/pdf/2021_March_JoI_Open_Source_Drives_Digital_Twin_SA.pdf

[40] (Jul. 2021). *Apache HTTP Server*. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Apache_HTTP_Server&oldid=1035340123

[41] (Jul. 2021). *Nginx*. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Nginx&oldid=1032222228

[42] *GitHub Pages*. Accessed: Jul. 26, 2021. [Online]. Available: https://pages.github.com/

[43] *Netlify*. Accessed: Jul. 26, 2021. [Online]. Available: https://www.netlify.com/

[44] *Azure Digital Twins*. Accessed: Jul. 26, 2021. [Online]. Available: https://azure.microsoft.com/en-us/services/digital-twins/

[45] *Azure Digital Twins Explorer*. Accessed: Jul. 26, 2021. [Online]. Available: https://docs.microsoft.com/en-us/samples/azure-samples/digital-twins-explorer/digital-twins-explorer/

[46] *Products—E-Magic TwinWorX Software Executive Summary*. Accessed: Jul. 26, 2021. [Online]. Available: https://e-magic.ca/products/

[47] *Mindsphere City Graph Software*. Accessed: Jul. 26, 2021. [Online]. Available: https://www.siemens-advanta.com/cases/mindsphere-city-graph

[48] Contributors. (Jul. 2021). *Admin-Shell-Io/Aasx-Package-Explorer*. [Online]. Available: https://github.com/admin-shell-io/aasx-package-explorer

[49] Contributors. (Jul. 2021). *Admin-Shell-Io/Aasx-Server*. [Online]. Available: https://github.com/admin-shell-io/aasx-server

[50] *Thingweb*. Accessed: Jul. 26, 2021. [Online]. Available: http://www.thingweb.io/

[51] Contributors. (Jul. 2021). *Eclipse/Editdor*. [Online]. Available: https://github.com/eclipse/editdor

[52] A. G. Mangas. (Jul. 2021). *Agmangas/Wot-Py*. [Online]. Available: https://github.com/agmangas/wot-py

[53] E. Korkan, H. B. Hassine, V. E. Schlott, S. Käbisch, and S. Steinhorst, "WoTify: A platform to bring Web of Things to your devices," 2019, *arXiv:1909.03296*. [Online]. Available: https://arxiv.org/abs/1909.03296

[54] *Evrythng*. Accessed: Jul. 26, 2021. [Online]. Available: https://evrythng.com/

[55] GS1. (2021). *Digital Link—Standards GS1*. [Online]. Available: https://www.gs1.org/standards/gs1-digital-link

[56] J. Autiosalo, R. Ala-Laurinaho, J. Mattila, M. Valtonen, V. Peltoranta, and K. Tammi, "Towards integrated digital twins for industrial products: Case study on an overhead crane," *Appl. Sci.*, vol. 11, no. 2, p. 683, Jan. 2021, doi: 10.3390/app11020683.

[57] R. Ala-Laurinaho, A. Nikander, and J. Autiosalo. (May 2021). *AaltoIIC/DT-Document*. [Online]. Available: https://github.com/AaltoIIC/dt-document

[58] D. Stolee. (Dec. 2020). *Get Speed With Partial Clone and Shallow Clone*. [Online]. Available: https://github.blog/2020-12-21-get-up-to-speed-with-partial-clone-and-shallow-clone/

[59] *Travis CI*. Accessed: Jul. 26, 2021. [Online]. Available: https://travis-ci.com/

[60] *GitHub Actions*. Accessed: Jul. 26, 2021. [Online]. Available: https://github.com/features/actions

[61] *GitLab CI/CD*. Accessed: Jul. 26, 2021. [Online]. Available: https://docs.gitlab.com/ee/ci/

[62] M. R. Crusoe, S. Abeln, A. Iosup, P. Amstutz, J. Chilton, N. Tijaniá, H. Ménager, S. Soiland-Reyes, B. Gavrilovic, and C. Goble, "Methods included: Standardizing computational reuse and portability with the common workflow language," 2021, *arXiv:2105.07028*. [Online]. Available: http://arxiv.org/abs/2105.07028

[63] E. Bouças. (Sep. 2017). *Creating a Static API from a Repository*. [Online]. Available: https://css-tricks.com/creating-static-api-repository/

[64] (Mar. 2021). *List of DNS Record Types*. [Online]. Available: https://en.wikipedia.org/w/index.php?title=List_of_DNS_record_types&oldid=1012819785

[65] A. Mannegal. (Feb. 2021). *Digital Twin Model Identifier*. [Online]. Available: https://github.com/Azure/digital-twin-model-identifier

[66] *The Official YAML*. Accessed: Jul. 26, 2021. [Online]. Available: https://yaml.org/

[67] W3C Permanent Identifier Community Group. (2021). *Permanent Identifiers for the Web*. [Online]. Available: https://w3id.org/

[68] Rebrandly. (2021). *Rebrandly Custom URL Shortener, Branded Link Management, API*. [Online]. Available: https://rebrandly.com/

[69] (Jan. 2021). *Violin Plot*. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Violin_plot&oldid=1000379419
[70] L. Sciullo, C. Aguzzi, M. Di Felice, and T. S. Cinotti, "WoT store: Enabling things and applications discovery for the W3C web of things," in *Proc. 16th IEEE Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2019, pp. 1–8, doi: 10.1109/CCNC.2019.8651786.
[71] L. Ding, T. Finin, A. Joshi, R. Pan, R. S. Cost, Y. Peng, P. Reddivari, V. Doshi, and J. Sachs, "Swoogle: A search and metadata engine for the semantic web," in *Proc. 13th ACM Int. Conf. Inf. Knowl. Manage.*, New York, NY, USA, Nov. 2004, pp. 652–659, doi: 10.1145/1031171.1031289.
[72] *JSON Schema*. Accessed: Jul. 26, 2021. [Online]. Available: https://json-schema.org/
[73] *ACE—The High Performance Code Editor for the Web*. Accessed: Jul. 26, 2021. [Online]. Available: https://ace.c9.io/
[74] J. D. Jong. *JSON Editor Online*. Accessed: Jul. 26, 2021. [Online]. Available: http://jsoneditoronline.org
[75] *YAML—Visual Studio Marketplace*. Accessed: Jul. 26, 2021. [Online]. Available: https://marketplace.visualstudio.com/items?itemName=redhat.vscode-yaml
[76] T. Anderson. (Mar. 2021). *PHP Repository Moved to GitHub After Malicious Code Inserted Under Creator Rasmus Lerdorf's Name*. Accessed: Mar. 31, 2021. [Online]. Available: https://www.theregister.com/2021/03/29/php_repository_infected/
[77] (Jul. 2021). *Query String*. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Query_string&oldid=1031686388

JUUSO AUTIOSALO received the B.Sc. and M.Sc. degrees from Aalto University, Espoo, Finland, in 2015 and 2017, respectively, where he is currently pursuing the Ph.D. degree. He was a Visiting Research Scholar with Michigan State University, in 2020. He has instructed six master's theses and two bachelor's theses and performed teaching activities at three university courses. He was the Project Manager of the DigiTwin project 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.

JOSHUA SIEGEL received the Ph.D., S.M., and S.B. degrees in mechanical engineering from the Massachusetts Institute of Technology (MIT). He is currently an Assistant Professor in computer science and engineering with Michigan State University and the Lead Instructor of the MIT, "Internet of Things" and "DeepTech" Bootcamps. He and his automotive companies have been recognized with accolades including the Lemelson-MIT Student Prize and the MassIT Government Innovation Prize. He has multiple issued patents, published in top scholarly venues, and been featured in popular media. His ongoing research develops architectures for secure and efficient connectivity, applications for pervasive sensing, including to vehicle diagnostics and new approaches to automated driving.

KARI TAMMI received the M.Sc., Lic.Sc., and D.Sc. degrees from the Helsinki University of Technology, in 1999, 2003, and 2007, respectively. He received teaching pedagogical qualifications with the Häme University of Applied Sciences, in 2017. He has been working as an Associate Professor with Aalto University, since 2015. He also works the Finnish Administrative Supreme Court as a Chief Engineer Counselor. Earlier, he worked as a Research Professor, the Research Manager, and the Team Leader and in other positions with the VTT Technical Research Centre of Finland, from 2000 to 2015. He has been a Postdoctoral Researcher with North Carolina State University, USA, from 2007 to 2008, and a Researcher at CERN, the European Organization for Nuclear Research, in 2000. He has authored more than 90 peer-reviewed publications cited in more than 5000 other publications. He currently serves as the Deputy Chair for IFTOMM Finland.