A Digital Twin Architecture for the Provisioning, Management, and Monitoring of Heterogenous IoT Devices

Jean-Marc J Maree
Department of Mechanical and Mechatronic Engineering
University of Stellenbosch
Stellenbosch, 7600, South Africa
jeanmarcmaree@pm.me

Karel Kruger
Department of Mechanical and Mechatronic Engineering
University of Stellenbosch
Stellenbosch, 7600, South Africa
kkruger@sun.ac.za

Anton H Basson
Department of Mechanical and Mechatronic Engineering
University of Stellenbosch
Stellenbosch, 7600, South Africa
ahb@sun.ac.za

ABSTRACT

IoT devices must often be configured to be remotely managed and monitored. The configuration process is referred to as a provisioning process, where the device is configured into an operational state. This paper presents a DT architecture for provisioning, managing, and monitoring heterogeneous IoT devices. The architecture is targeted towards the *device-as-a-service* (DaaS) context, where the complexities involved in managing IoT devices is hidden from the user, and where the uptime of the IoT devices is critical. Device management here includes installing applications on the devices (to fulfill the functional requirements of the user) and monitoring the devices. The architecture presented here is aimed at use in a large manufacturing enterprise, such as an automotive manufacturer. A preliminary evaluation of the architecture shows its potential to dynamically handle the provisioning, managing, and monitoring of IoT devices. The preliminary evaluation shows how the architecture supports devices that use both publish-subscribe and request-response interaction methods, and even handle complex scenarios containing a human in the loop.

CCS CONCEPTS

• **Software and its engineering** → Software organization and properties; *Software system structures; Abstraction, modeling and modularity*.

KEYWORDS

Digital twin, IoT device management, IoT device provisioning

MODELS Companion '24, September 22–27, 2024, Linz, Austria

ACM ISBN 979-8-4007-0622-6/24/09.
https://doi.org/10.1145/3652620.3688258

ACM Reference format:
Jean-Marc J Maree, Karel Kruger and Anton H Basson. 2024. A Digital Twin Architecture for the Provisioning, Management, and Monitoring of Heterogeneous Iot Devices. In *ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems (MODELS Companion '24), September 22–27, 2024, Linz, Austria.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3652620.3688258

1 INTRODUCTION

The Internet of Things (IoT) is a term used to describe the interconnection between physical and virtual entities, through network connections [1, 4]. The physical entity, i.e. the IoT device, has physical attributes that provide services of sensing or actuating. The virtual entity represents digital characteristics of the physical device, controlling or monitoring the physical entity from a remote environment – typically the cloud. IoT devices are key enablers for industry 4.0 [5, 8] and are attractive for use in a device-as-a-service (DaaS) model to serve wider enterprise requirements.

DaaS is the service of outsourcing the configuration and maintenance of hardware and software systems to a DaaS provider. The aim of DaaS is to reduce, and sometimes remove, the workload of an enterprise's IT team and to allow enterprises to scale their utilized devices according to the demand [14, 17, 22].

The DaaS provider must manage the entire lifecycle of heterogeneous devices, including its *provisioning*, operation with remote *management* and *monitoring*, and *replacement* or *decommissioning*. The DaaS provider must also provide diagnostics and support to the device's end user, to avoid impeding on the subscribed users' enterprise operations. [22]

An overarching challenge in working with IoT devices is their heterogeneity in functionality, communication technologies, implemented applications, and message structures [4, 9, 20, 30, 31]. This is because there are no standards that the devices must adhere to before commercialization [4, 30, 31]. Remote management of IoT devices is essential in an industrial setting, as the downtime of a device may affect an enterprise's operational efficiency. When enterprises introduce heterogeneous IoT devices into their workflows, their development teams often first have to develop or adapt existing IoT device management platforms.

Only then the new types of IoT devices can be deployed for their intended use case.

Digital twins (DTs) have been widely used to represent complex physical products throughout their lifecycle. In such cases, the DTs have to interact with heterogeneous data sources and make provision for bi-directional interactions between the DT and its physical twin (PT). DTs offer the potential to manage IoT devices throughout their life cycle, specifically in a DaaS context that requires an accurate representation of the devices' status, using heterogeneous data sources.

This paper presents a DT architecture for an IoT DaaS platform that facilitates the provisioning, management, and monitoring of heterogenous IoT devices. Section 2 reviews related IoT work, the characteristics of a DT, and the potential value DTs present to IoT. Section 3 presents the requirements of the DaaS platform. Section 4 presents the architecture and its components, while Section 5 offers further details on the internal architecture of the DT. Section 6 gives an illustrative example of the platform, leading to a discussion of the overall architecture in Section 7. Section 8 concludes the paper, with recommendations for future work.

2 RELATED WORK

2.1 Digital Twins and their Characteristics

Digital twins (DTs) integrate physical systems with their virtual representations to facilitate monitoring, analysis and control. This characteristic has led to DTs being widely used in industrial system applications for condition monitoring, modeling and simulation, predictive analytics, optimization, and decision-making [29]. The digital twin (DT) was first introduced by [12] in a 2003 presentation on product lifecycle management. At the time, the technologies enabling DTs were limited, but have since evolved, leading to a wide use of DTs in manufacturing and further research into the field of DTs.

A comprehensive review of the characteristics and capabilities of a DT is beyond the scope of this paper, but researchers have outlined the following core characteristics of DTs [21, 24, 32, 36]:

- *Dynamic*: The DT and PT over time experience a bi-directional change in state and reported measurements. The integration of data between the two is continuous and makes provision for a human in the loop, where necessary.
- *High-fidelity*: The DT must represent an accurate model of the PT, using characteristics that contribute value to reflecting the real state of the PT.
- *Self-evolving*: The DT associates itself with the PT through various stages of the PT's life cycle. The DT can adapt and optimize the performance of its PT through simulation, emulation and feedback control.
- *Hierarchical*: The DT can represent a component within a larger system, and the hierarchy of DTs can be used to represent a large, complex system.

The characteristics of DTs has led it to be used in large complex systems, resulting research in architectures that serve as an outline that a software system may be built upon [24]. One such architecture for complex systems is the highly adaptable service-oriented architecture (SOA) for DTs, shown in Figure 1.

The architecture contains *management services*, a *service network*, a *gateway service*, an *aggregation hierarchy*, and a *central user interface.* The management services are used to maintain the overall operation of the system and its DTs, while the service network contains services that are used to fulfill specific user requirements or are shared among DTs. The gateway service serves as the entry point for external software systems and user interactions.

The DTs in the aggregation hierarchy each contains its own data, models, and services. The DT services differ from those in the services network with the DT services relying only on the data of their DT, while the services network services can use data from multiple DTs and external data sources. The aggregation hierarchy makes provision for digital twin instances (DTIs) and digital twin aggregates (DTAs). A DTI represents a unique instance of a PT, while a DTA represents a more complex PT by grouping multiple DTIs and/or DTAs. [35] expands on the DTA concept by introducing the type digital twin aggregate (TDTA), a grouping of standardized representations of multiple DTIs. [35] argues that the TDTA allows for batch processing instead of concurrent similar DTIs performing the same task.

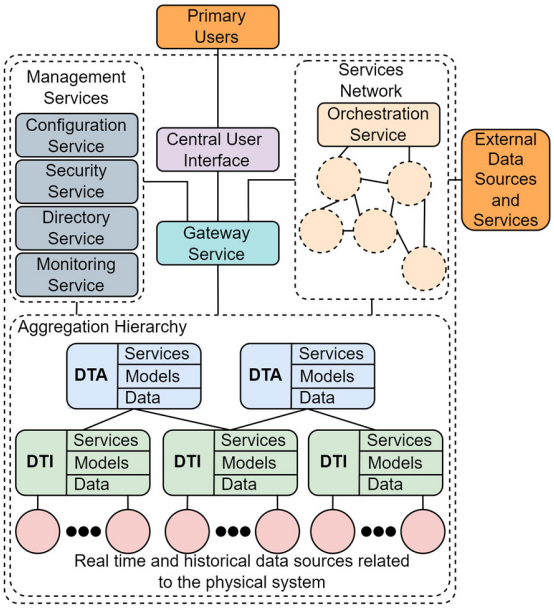

Figure 1: Reference architecture for digital twins [15] [© Stellenbosch University; used with permission]

2.2 IoT Device-as-a-Service

Modern IoT systems leverage cloud computing and SOA by reusing existing services in the cloud to rapidly meet an enterprise's requirements. The SOA-based IoT architecture has four layers:

- *Sensing layer:* The physical layer, where the device interacts with the environment through a sensor or an actuator.
- *Network layer:* The transport layer that connects the physical device to the cloud.
- *Service layer:* The middleware layer that contains services that are discoverable, interfaceable via an API, trustworthy, and are used to fulfill functional logic.
- *Interface layer:* The layer that external systems and UIs can interact with.

Cloud service providers, such as Amazon Web Services [37], offer the opportunity to connect IoT devices to cloud services through an IoT gateway – an entry point for specific IoT technologies [37]. Once the device is connected, its messages are forwarded to appropriate subscribers (if using a publish-subscribe model) or server (if using a request-response model).

In a DaaS platform, the platform managers must manage the different life cycle phases of the IoT devices. The core phases of the device's life cycle are [28, 33, 38]:

- *Development*: In addition to adapting the device technology to interact with a management platform's interfaces, the development process includes using metrics to accurately represent the device's current status [34].
- *Onboarding*: The first step after acquiring a device is to establish its initial ownership [26]. The onboarding process involves transferring initial credentials to the device and configuring an initial virtual entity that will be associated with the device [33, 37, 38]. Additionally, preliminary services, which maintain interfaces for interacting with the *provisioning* server, are installed on the device. The initial credentials are used so that the device can establish connection with the *provisioning* server [2, 37].
- *Provisioning*: The process of provisioning includes the creation of unique credentials, configurations, and security policies. This information is associated with the device's virtual entity, some of which, such as the credentials, are transferred to the device [2]. The provisioning process can either be performed manually, semi-automatically (with human intervention), fully automatically, or even with a human in the loop [37].
- *Operation*: The operation phase of the device includes the *management* and *monitoring* of the device [38]. The management of the device includes remote configuration, where apps can be remotely installed on the device and its state can be controlled. Monitoring is the ingestion of the device's resource metrics and the status logs of the installed apps. Using rules and/or models, the ingested metrics and logs are used to infer the state of the device.

2.3 Digital Twins in IoT

The use of DTs to manage IoT devices has been hinted at by open-source software projects. FIWARE [11], Eclipse Ditto [10], and Azure IoT Hub [3] – popular services that simplify the development of IoT systems dependent on heterogeneous IoT technologies – deploy DTs that are hierarchical, dynamic and of high-fidelity. However, they lack the self-evolving nature of DTs and rely on external monitoring services for this functionality. [7] argues that the approach of relying on only data passed to external monitoring services, constrains the opportunity for context-aware service executions, as only simple contextual tags are passed to the monitoring services.

3 REQUIREMENTS FOR AN IOT DEVICE-AS-A-SERVICE PLATFORM

The functional requirements (FRs) and key non-functional requirements (NFRs) for a heterogeneous IoT DaaS platform in an industrial context are outlined in Table 1 and Table 2. The background to these requirements is given in a companion paper [23]. Table 1 outlines the FR, with a description and the phase of the IoT device's lifecycle that is directly affected by the requirement. The NFRs in Table 2 were established from existing DaaS platforms, as well as from standards [18, 19]. The exhaustive formulation of NFRs for such an application is beyond the scope of this paper. Only NFRs deemed to be application independent and those relevant to the discussion in Section 7 are listed in Table 2

It should be noted that this paper assumes that the devices use open software without.

4 ARCHITECTURE

This section presents a DT architecture for the provisioning, management, and monitoring of heterogeneous IoT devices in a DaaS context. Section 4.1 presents an overview of the architecture, and the remainder of this section introduces the components of the architecture.

The characteristics of DTs make them suitable for the provisioning, management, and monitoring of IoT devices because the DT will retain a dynamic, high-fidelity representation of the IoT device's state and capabilities. Additionally, the DT can use services, models, and data to adapt to the IoT device's ever-changing conditions and the hierarchal characteristics of the DT can be used to manage groups of IoT device. Lastly, the characteristics of DTs support implementations that ensure that NFRs of complex IoT systems are met [23].

Table 1: Functional requirements for IoT device management platforms

ID	Description	IoT Device Lifecycle Phase
FR1	Provision devices manually, semi-automatically, or fully-automatically.	Onboarding; Provisioning
FR2	Interact with devices of heterogenous technologies, capabilities, computational resources and operating system (OS) architecture.	Development; Provisioning; Operation
FR3	Monitor metrics and application logs reported by the device, and provide the history of this data to external analysis services.	Operation
FR4	Generate alerts if metrics have exceeded user defined thresholds.	Operation
FR5	Remotely manage the state and configuration of the device.	Operation
FR6	Provide interface for users to remotely manage the apps installed on the device.	Operation
FR7	Provide interfaces for device owners, platform configurators and maintainers to interact with the platform. Further provide interfaces for external applications to ingest their device's data.	Development; Provisioning; Operation
FR8	Convert different IoT device protocols, to a standard platform protocol.	Development; Provisioning; Operation
FR9	Ensure peripheral and software compatibility between the device and available apps.	Operation
FR10	Provide the device's contextual data when making decisions for managing the device. Contextual data includes information pertaining to location, time and sequence of executed events.	Provisioning; Operation

4.1 Overview

Figure 2 presents the DT architecture for the DaaS platform, where heterogeneous IoT devices can be managed by the platform. The diagrams are based on the C4 model [6].

Figure 2 shows different user interfaces for the provisioning, management, and monitoring processes, with corresponding operators. Furthermore, the architecture makes provision for external data destinations so that external parties can access their device's metrics.

Table 2: Non-functional requirements for IoT device management platform

ID	Name	Description
NFR1	Flexibility	Support integrating new device technologies, and devices of varying capabilities, while limiting the propagation of changes throughout the platform.
NFR2	Security	Authenticate and authorize the device's access to the platform and its internal services.
NFR3	Discoverability	Support service and capability advertisement, allowing system components to dynamically select services that fulfill their requirements.
NFR4	Integrability	Integrate with IoT devices of varying protocols, OS, resource capabilities, and functional capabilities.
NFR5	Legacy Support	Able to integrate with existing IoT devices.

The key services in the platform are grouped into four categories:

- *Platform management services*: Services used to monitor the high-level status of the platform, and the users and apps associated with the platform.
- *DT management services*: Services that manage and control the state of the DTs, including their authentication identity, authorization policies, and data.
- *Device management services*: Services that directly affect the operational state of the device, and its interactions with its associated DT.
- *Service network*: Common, loosely-coupled services shared between different DTs, and services that are used to fulfill functional requirements of the DT.

These key services collaborate with the *DT hierarchy*, *transformer* and *IoT gateway* to enable interactions with an *IoT device client*. Lastly, interfaces and external systems can interact with the system through a *user gateway service*. A discussion of the components of the architecture follows next.

4.2 IoT Platform Management Services

4.2.1 User Identity Manager. This component is responsible for holding the user's profile and associating the user's profile with a group. The group can represent a separation of departments or development phases within the department. The DTs are associated with the groups, and the authorization of the groups to specific DTs is decoupled from the user identity manager.

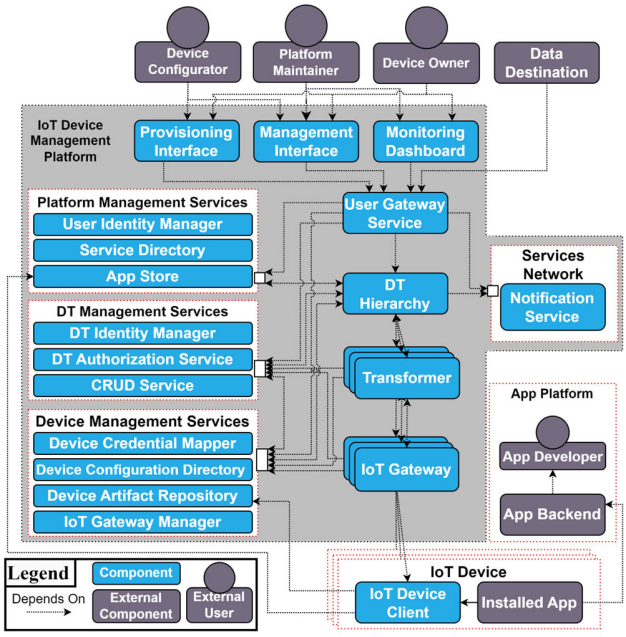

Figure 2: DT architecture for IoT DaaS [© Stellenbosch University; used with permission]

4.2.2 Service Directory. The service directory is a global directory where services and DTs can advertise their capabilities, interfaces, and entry points for other services and DTs to use. The service directory component decouples services reliant on request-response communication, by allowing for dynamic discovery and selection processes. If a service has changed, it would update its entry in the service directory and other services would query the updated changes.

4.2.3 App Store. The app store is a global index of the available apps on the platform, where app versions are not limited to a single device technology. The detailed operation of the app release cycles is beyond the scope of this paper, but the requirement is that an app is maintained by an independent party. The app that the independent development team maintains may have different versions developed for different OSs or IoT device technologies. Regardless, it is the responsibility of the app development team to reflect the app version with the required compatible OS, device capabilities, and the required connected peripherals.

The app store may connect to external repositories that store the compiled apps and will associate the access point of these apps with their version number and metadata indicating compatibility requirements.

4.3 DT Management Services

4.3.1 DT Identity Manager. The DT Identity Manager is the global identity manager for all DTs within the platform. This service contains a unique identifier alongside the MAC address, model, and serial number of the device. This service only retains static information that can be used to identify the device, while minimizing the use of the device's information to avoid data duplication.

4.3.2 DT Authorization Services. The authorization service maintains the accessibility of DTs to different services, users to DTs, and services to DTs. The authorization service is globally accessible and will validate if a particular requester has read and/or write access to the requested DT or service. The authorization model of choice is an implementation decision governed by platform requirements, but for provisioned DTs, the DTs at higher levels in the hierarchy are responsible for facilitating the appropriate authorization of their respective DTIs.

4.3.3 CRUD Service. All create, read, update, and delete (CRUD) operations on DT data are performed through the CRUD service. For read requests, the CRUD service checks the authorization of the request and returns the results that the requester is authorized to view. Create, update, and delete requests may be long-running jobs, which the CRUD service can asynchronously facilitate without retaining a long-running connection with the requester. The CRUD service is also responsible for mapping a user's request to their associated DT, in collaboration with the authorization service.

4.4 Device Management Services

4.4.1 Credential Mapper. The credential mapper associates the DT ID with the associated IoT device credentials. These credentials may include an API key and combinations of a username, a password, and certificates. Access to these credentials should be strictly controlled to prevent leakage of the credentials, as these credentials may be used to masquerade the device's connection to the platform.

4.4.2 Device Configuration Directory. The device configuration directory serves as a repository of configurations and templates that must be populated. As new device technologies are added to the platform, their associated configurations and templates are stored in the configuration directory so that the DT can:

- Request and store a configuration on the device.
- Request a template, populate and store the populated template in the DT, and store the populated template on the device.

4.4.3 Device Artifact Repository. The device artifact repository stores all software packages and dependencies required for the device to interact with the management platform. The repository retains software packages, including the OS that may be flashed onto the device, and the IoT device

client that is installed onto the device. This repository maintains artifacts for heterogeneous devices and the artifacts are downloadable by services outside of the platform.

4.4.4 IoT Gateway Manager. This is a centralized manager that interacts with the IoT gateways used by different device technologies. The IoT gateway manager facilitates the process of adding and/or removing a device's access from their specific IoT gateway. Separate gateways may be used for the provisioning and operational phases of the device, and the IoT gateway will facilitate the transition of the device's access from one gateway to another.

4.5 Service Network

At present, the alert service is the only service in the service network, but other services could be added to fulfill the functional requirements of the DTs. The DTs and other services can generate a notification request, because of a triggered event, and the notification request will inform appropriate staff members of the event. The notification may be in the form of an email, a support ticket, or an SMS, depending on the implementation and the severity of the event.

4.6 IoT Device Client

The IoT device client is the application on the device that interacts with the DT through the IoT gateway and transformer. The IoT client performs the following actions:

- Facilitates app installations from the App Store onto the device.
- Measures and publishes device resource usage to the DT.
- Collects the status logs of the installed apps and publishes the logs to the DT.

For full remote management, the IoT device client requires elevated access to the device's system APIs so that it can install apps onto the device without user intervention. The elevated access may grant further opportunities for controlling the device's state.

4.7 User Gateway Service

The gateway service acts as an entry point to the platform and performs the task of load balancing traffic to available software services. The gateway service can also perform the function of basic authentication, encrypting network traffic between the gateway service and external users, and hide the internal addresses of the DT system's components, routing a user's request to a specific address to the appropriate service.

4.8 DT Hierarchy

The DT hierarchy contains the DTIs that represent the IoT devices and the TDTAs that represent a type grouping IoT devices. The type grouping is applied to IoT devices that share the same model, communication methodologies, and provisioning requirements. The TDTA takes ownership of its associated DTIs, including being responsible for provisioning the DTI and ensuring that the DTI can interact with its associated device and vice versa.

The DT hierarchy makes provision for DTAs that aggregate DTIs to represent a complex physical system. A DTA's configurations can also be applied to its DTIs, e.g. a DTA aggregating all the devices in a building can convey the inactive hours of the building to its DTIs so that all the devices can install app updates during those hours. Figure 3 presents an example of the relationship between the DTI, TDTA, and DTA, where a DTI is maintained by only a single TDTA and the DTI can inherit configurations from multiple DTAs with which it is associated. The ability for DTIs to inherit properties from their associated DTAs allows for bulk customization of a DTIs, with more refinement than the type grouping of the TDTA. Notably, the ability of the DTA to change the configuration of a DTI presents complexity, further discussed in Section 7.

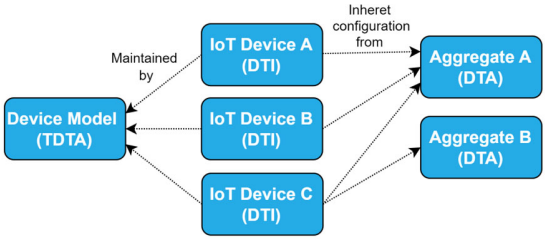

Figure 3: DT hierarchy example [© Stellenbosch University; used with permission]

4.9 Transformer

The transformer's internal architecture, as shown in Figure 4, converts data formats from the IoT protocol into a DT standard and vice versa. The transformer contains a client and a server for the internal protocol that interacts with the DTs, and a client and a server for the IoT protocol that interacts with the IoT device through the IoT gateway.

The data transformer is the component that performs the data conversion and maps the IoT device identifier to the DT identifier, with the use of the credential mapper. Assuming that the interaction between the credential mapper and the data transformer is request-response, provision is made for service discovery by querying the service directory for the credential mappers accessible address. Furthermore, to avoid continuous queries to the credential mapper, the data transformer can store the device's reference mapped to the DTs reference in cache.

It should be noted that the transformer does not access the authentication credentials of the IoT device. Regardless of whether the IoT devices communication is based on request-response or publish-subscribe, the transformer hides the communication protocol from the DT.

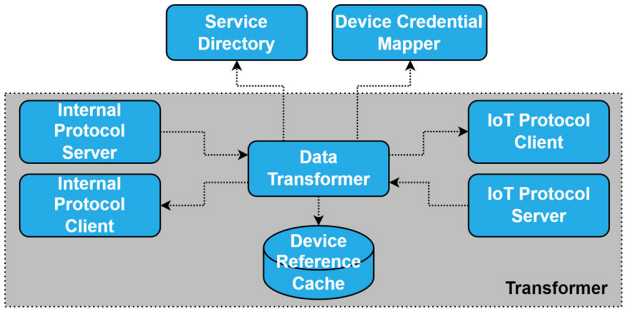

Figure 4: Transformer internal architecture [© Stellenbosch University; used with permission]

4.10 IoT Gateway

The IoT gateway is the communication entry point into the platform for multiple IoT devices. The components of the gateway, shown in Figure 5, includes a communication mediator, a gateway configurator, and the persistent storage for the gateway configuration. The gateway configurator is a server that listens to requests from the IoT gateway manager, which will request the addition or removal of devices to or from the IoT gateway. The communication mediator represents systems such as an MQTT broker [25], and contains its own authentication system. When the gateway configurator registers a device to the communication mediator, the result (containing the IoT device's credentials) is sent to the IoT gateway manager which stores the credentials in the device credential mapper. The result is that the IoT device's credentials are only stored in the device credential mapper and in the communication mediator, which maintains the credentials.

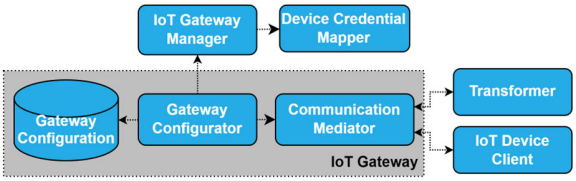

Figure 5: IoT gateway internal architecture [© Stellenbosch University; used with permission]

5 DIGITAL TWIN INTERNAL ARCHITECTURE

The internal architecture of a TDTA, which manages the provisioning and operation phases of a device, is shown in **Figure 6**. This DT's architecture was selected for discussion as it is the most complex DT architecture within the platform – the DTI and DTA have similar, but simpler, internal architectures. The discussion of the internal architecture will focus on how the TDTA uses data, models, and services to maintain the IoT device.

The *message handler* routes all incoming messages into the DT to the appropriate service.

The *DT CRUD service* performs the following actions:

- Create: Creating a new DTI, when a new device is added to be managed by the platform.
- Update: Updating the DTI's metadata, including its associated configuration and apps.
- Read: Reading the metadata of the DTI, the DTI's current provisioning status, and/or its historic metadata.
- Delete: Removing historic data from the DTI, removing associated configurations from the DTI, and/or placing the DTI in an archived state.

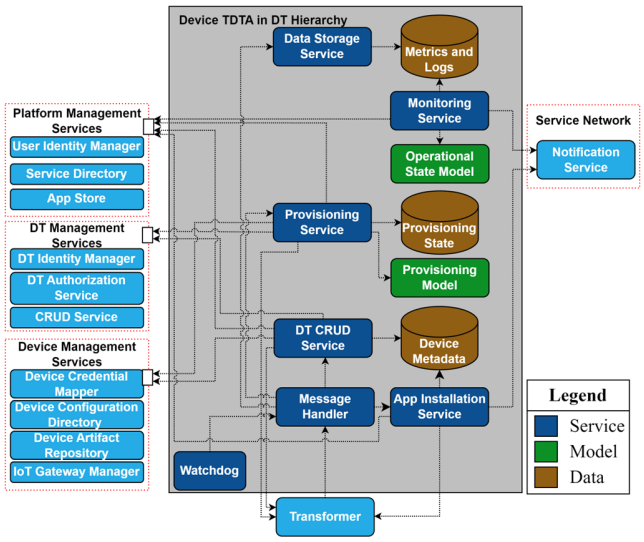

Figure 6: Internal architecture of IoT Device TDTA [© Stellenbosch University; used with permission]

The *provisioning service* orchestrates the provisioning of DTIs of the type associated with the TDTA. The provisioning process uses a state machine model of the provisioning process. Therefore, alterations in the provisioning process require an adaptation in the provisioning model.

The *data storage service* receives and stores the device's metrics and log data in their appropriate locations. The *monitoring service* ingests the data stream and infers the state of the device, using a rule-based model of the device's operational state. If the monitoring service infers that an operational status is outside of typical conditions, or that a status exceeds defined thresholds, the monitoring service will trigger an event to the notification service, which then informs relevant parties of the abnormal operating conditions.

The *app installation service* interacts with the App Store, receiving updates when new app versions are available. The app installation service will validate the compatibility of the app with the device's metadata, before creating a request to install a new app version to the device.

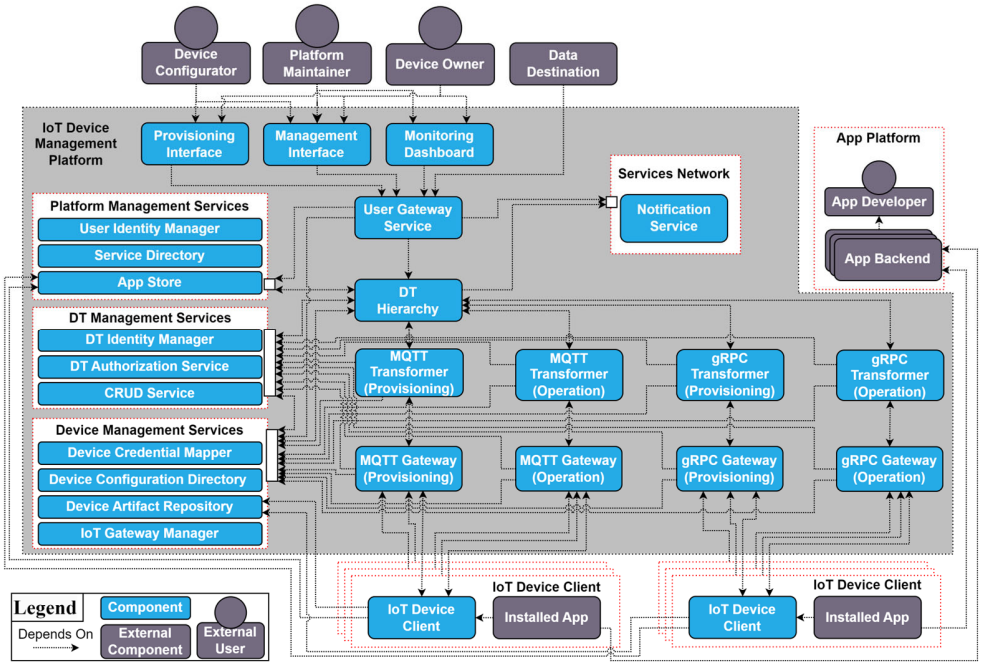

Figure 7: Use case example of DT-based IoT DaaS platform [© Stellenbosch University; used with permission]

If a manual approval is required for the app installation, the app installation service can notify the owner of the IoT device of the update. Once the installation has been approved, the app installation service will send the app installation request to the device via the transformer.

Lastly, the TDTA has its own *watchdog service* to monitor whether services are running longer than expected. The watchdog interacts with the CRUD service, the provisioning service, and the data monitoring service, which are all services that may rely on external asynchronous operations. If the watchdog service's wait-time for a response from asynchronous service providers exceeds a defined threshold, the watchdog will either initiate a query to the service (where the service can retry the request) or generate an alert event to the notification service.

6 ILLUSTRATIVE EXAMPLE

An illustrative example of the above architecture is shown in Figure 7.

The example system makes provision for IoT devices using MQTT [25], and gRPC [13]. The use of these two protocols shows that the architecture works with widely used technologies, and that provision is made for both publish-subscribe and request-response interaction methods.

Furthermore, the illustrative example uses two separate IoT gateways for the provisioning and operational phase of the device, and the example uses a separate transformer for the different gateways. To reduce the number of operational components to maintain, a single transformer may alternatively be used, but the transformer will contain separate IoT clients used to communicate with the IoT gateways.

The illustrative example shows horizontal scalability and the operation of existing components and DTs are not affected when introducing new managed device technologies.

7 DISCUSSION

This section reflects on the design decisions and considerations during the development of the system architecture, and how different system components contribute to meeting individual FRs and NFRs.

7.1 User Interaction

Provision is made for different users (device configurator, device owner, and platform maintainer) to interact with interfaces to provision, manage and monitor the IoT device's life cycle (FR3, FR5, FR7). Furthermore, the architecture supports the integration with a user's external data destination (FR3 and FR7).

To ensure that a user only has authorization to access their own DTs, a key design decision was to decouple the user's identity stored in the user identity manager from the DT hierarchy and other components in the architecture. The DT authorization service stores the authorization relationship between DTs and other system components, including the DTs and their associated user identity. All architectural components within the IoT management platform have access to the DT authorization service and can verify the authorization of the source of a request (NFR2). The use of the CRUD service presents an opportunity for a single service to

facilitate a user's interaction with the DT. The CRUD service can handle the request made by a user, including authorizing the request and interacting with the appropriate DT. Furthermore, the CRUD service can initiate long-running requests where no immediate result is available.

7.2 Integrating with Diverse IoT Technologies

The transformer and the IoT gateway are the two components responsible for fulfilling FR2 and FR8, while making provision for NFR2, NFR4, and NFR5. A design decision was to separate the two components to simplify the internal logic of the individual components. Separating the two components allows for the separation of the provisioning phase's network traffic from that of the operational phase. This separation ensures that only authorized traffic is being processed by the operational phase's transformer and the component's interfaces that it interacts with.

Alternatively, a single transformer can interact with the IoT gateway for both the provisioning and operational phases, by using the service directory and device credential mapper to authorize and route the interactions to the appropriate DT interfaces. However, this may not be suited for large scale environments, as the authentication requirements for the provisioning IoT gateway are less stringent than that of the operational IoT gateway (i.e. generic credentials may be used during the provisioning process). This lowered stringency opens the opportunity of saturating the provisioning transformer with undesired or even unauthorized requests, affecting the operational phase of the IoT devices if the transformer of the phases is combined.

IoT devices use the IoT gateway as an entry point for their communication and the transformer interacts with the IoT device through the gateway. These interactions use an IoT protocol client and a server that work for both request-response, where a client-server interaction takes place, and publish-subscribe, where clients connect to a message broker.

The IoT gateway configurator, in collaboration with the IoT gateway manager, hides the nuances of adding devices to the platform from the internal components. The decision was made to allow the TDTA to register the DTI with the IoT gateway manager using passed contextual information, e.g. the region. The IoT gateway manager could then assign the device to the geographically nearest IoT gateway. The benefit of this approach is that the gateway can hide how it assigns and balances the devices assigned to different gateways. However, it is acknowledged that following autonomous practices of the DT, the DT is also capable of searching for available IoT gateways and assigning itself to the appropriate IoT gateway.

The challenges the latter approach introduces is the potential of unauthorized DTs having undesirable read and/or write access to IoT gateways, and the potential instability of the IoT gateways connection to devices if there is no centralized oversight of gateways performance.

7.3 IoT Device Credential Handling

The IoT gateway manager may retain the reference of the DT IDs it has allocated to various gateways and retain the relationship between the DT ID and the associated IoT device reference, but the global index for this relationship is the device credential mapper service. Inspired by legacy integration mechanisms proposed by IBM [16] and Oracle [27], the credential mapper serves as a key-value pairing of the IoT device reference and the DT ID, and stores the authentication credentials that the IoT device may use to connect to the platform (NFR4, NFR5).

The device credentials are only required during the provisioning process of the device, when the device's unique credentials are created and transferred to the device. Once the credentials are stored on the device, and the provisioning process is complete, the credentials can be removed. The design decision to store the credentials in the device credential mapper is to make provision for the persistency of the credentials for different asynchronous provisioning processes. Instead of storing the credentials in the DT, the justification for an external service was focused on large software systems, where the domain knowledge in authentication systems would focus on the device credential mapper, while the DT domain knowledge could simply integrate with the existing service.

An alternative approach would be to merge the credential mapper with the DT identity center, forming a single service that reduces code complexity and data duplication. Future implementations may follow this option.

It should be noted that storing credentials anywhere in the platform presents risks and it is best practice to store the credentials in an encrypted format, where only authorized parties (the device) are in possession of the decryption keys and can receive the value, decrypt it, and use it.

7.4 Device Software and Configuration

The artifact repository, device configuration directory, and app store are intended to serve as data repositories that maintain interfaces that the DTs and other services can interact with. These components attach metadata to the stored data, which the DT can interpret. The attachment of metadata makes provision for the DT to dynamically search for appropriate configuration and software, and to choose the most appropriate configuration to apply or software to install (FR5, FR9, NFR1, NFR3 and NFR4).

For mission critical implementations, rigid rules may be more suitable and the DTs can index the exact configuration or software package they are looking for. The proposed architecture supports both static and dynamic indexing, and the opportunity that dynamically indexing package repositories presents is that the service can be dynamically adapted to changes, with minimal or no change in code. When a new type of DT (e.g. a new device model using an already supported OS) is introduced into the platform, it can reuse

existing configurations, software packages and/or apps.

Furthermore, to avoid data duplication, the DT can store custom configurations in its data and reference the data stored in the service, which it can query when required.

7.6 TDTA Role in Provisioning

There is an overlap between how the TDTA acts as the provisioning service and how other architectures use a provisioning server [37]. Both facilitate the provisioning process for a type of device technology, but using a TDTA as the provisioning server centralizes the contextualization of the data, with greater autonomy in selecting how it must provision the devices.

Fulfilling the provisioning process (FR1, NFR2) as a TDTA presents a means towards a scenario where DTs in fields beyond IoT devices can be dynamically provisioned and their physical twin can be automatically or semi-automatically configured during the provisioning process. Furthermore, a DT approach to provisioning devices can accommodate a combination of provisioning processes, including processes requiring a human-in-the-loop. A step-by-step log of these processes can be centrally stored within the DT. As the architecture currently stands, if a different provisioning process is required for a device of the same type, it will require a separate TDTA with a new model to orchestrate the process.

7.7 Device Management

The DT offers the opportunity to validate the compatibility of apps installed on the device using the metadata from the app store (FR6 and FR9). Validating the compatibility of apps is a trivial task for any system when managing a single type of device. However, as the amount of device heterogeneity increases, complexity increases too – especially when considering the backwards compatibility of available interfaces on OSs. The contextual data, models, and services within the DT enable it to internally handle heterogeneous requirements (FR6).

Collaboration characteristics of the DT also present an opportunity for devices reporting errors when an app has been installed. Then their DTs can report the issue to devices of the same type, who can then either roll-back or postpone the installation of the app version. Furthermore, the DT can use its simulation and emulation characteristics, together with the historic data to create models of the device's idle and operational hours, to perform actions including controlling the state of the device's power consumption (FR5).

7.8 Device Monitoring

For the monitoring of the device's state, the DT can either use a rule-based model or an ML-based model to infer the state of the device from its heterogenous data-sources. As data is ingested into the DT, the monitoring service inside the DT will infer the status and, if it is outside of defined thresholds, the DT will trigger an event to the notification service (FR3, FR4 and FR10).

As with the interface validation for the app installation process, this task increases in complexity with heterogeneity. Furthermore, devices of the same type can perform different functions, while different device types can perform the same functions. TDTAs can monitor the status of a single type of device, using general models or thresholds to define typical operating conditions. For more specialized monitoring of the device's status, a DTI can monitor the status of its pertaining device and use unique models or thresholds. Furthermore, a DTA can group DTIs performing the same function (e.g. devices in the same department), providing further refinement of the models or thresholds used to infer the device's status.

Either the DTI or the TDTA masquerading as the DTI will receive continuous data from the IoT device. The trade-off of such a decision is *performance* vs *cost*. The consideration for performance is that the number of devices being monitored may be a bottleneck if only a single TDTA is used to store data. If each DTI is running in separate compute instances, they will each incur an operational cost. However, if the scale can be handled by a TDTA, then only that DT will incur an operational cost.

8 Conclusions

This paper presents a DT architecture for an IoT DaaS platform that facilitates the provisioning, management, and monitoring of heterogeneous IoT devices. The context of the proposed IoT architecture is focused towards a DaaS platform, where the complexity of the IoT devices is abstracted from its end users, the assurance of uptime of the IoT device is critical, and the more IoT devices the platform can maintain, the more users it will attract. For large systems, maintaining thousands of IoT devices, the platform must make provision for automated and semi-automated processes to add the devices onto the platform.

The paper contributes to the research community by showing the value of combining the characteristics of the different types of DT and to use them in the different phases of an IoT device life cycle. The modularity of the system architecture abstracts the heterogeneity of IoT devices. The architecture presented here has the potential for significant practical value for the fine-grained life cycle management of heterogeneous IoT devices at a large scale.

Further work is needed to evaluate the performance of this architecture, using one or more case studies, and to improve the heterogeneity of TDTAs without duplicating the TDTA entirely. Furthermore, additional work is required in identifying how the DTIs should be grouped, while adhering to the authorization requirements and minimizing the number of compute instances required.

REFERENCES

[1] S. M. Abu Adnan Abir, Adnan Anwar, Jinho Choi, and A. S. M. Kayes. 2021. IoT-Enabled Smart Energy Grid: Applications and Challenges. *IEEE Access* 9, (2021), 50961–50981. https://doi.org/10.1109/ACCESS.2021.3067331

[2] Azure. 2022. What is Azure IoT Hub Device Provisioning Service? Retrieved May 1, 2023 from https://learn.microsoft.com/en-us/azure/iot-dps/about-iot-dps

[3] Azure. 2024. Azure IoT Hub. Retrieved January 17, 2024 from https://azure.microsoft.com/en-us/products/iot-hub/

[4] Sharu Bansal and Dilip Kumar. 2020. IoT Ecosystem: A Survey on Devices, Gateways, Operating Systems, Middleware and Communication. *Int J Wireless Inf Networks* 27, 3 (September 2020), 340–364. https://doi.org/10.1007/s10776-020-00483-7

[5] Laura Belli, Luca Davoli, Alice Medioli, Pier Luigi Marchini, and Gianluigi Ferrari. 2019. Toward Industry 4.0 With IoT: Optimizing Business Processes in an Evolving Manufacturing Factory. *Front. ICT* 6, (August 2019), 17. https://doi.org/10.3389/fict.2019.00017

[6] Simon Brown. 2016. The C4 model for visualising software architecture. *Simon Brown*. Retrieved April 28, 2024 from https://c4model.com/

[7] Roberto Casadei, Giancarlo Fortino, Danilo Pianini, Wilma Russo, Claudio Savaglio, and Mirko Viroli. 2019. Modelling and simulation of Opportunistic IoT Services with Aggregate Computing. *Future Generation Computer Systems* 91, (February 2019), 252–262. https://doi.org/10.1016/j.future.2018.09.005

[8] Ioannis T. Christou, Nikos Kefalakis, John K. Soldatos, and Angela-Maria Despotopoulou. 2022. End-to-end industrial IoT platform for Quality 4.0 applications. *Computers in Industry* 137, (May 2022), 103591. https://doi.org/10.1016/j.compind.2021.103591

[9] Alem Čolaković and Mesud Hadžialić. 2018. Internet of Things (IoT): A review of enabling technologies, challenges, and open research issues. *Computer Networks* 144, (October 2018), 17–39. https://doi.org/10.1016/j.comnet.2018.07.017

[10] Eclipse Foundation. 2023. Eclipse Ditto. *Eclipse Ditto*. Retrieved February 26, 2024 from https://projects.eclipse.org/projects/iot.ditto

[11] FIWARE. 2023. About FIWARE. *About FIWARE*. Retrieved May 19, 2023 from https://www.fiware.org/about-us/

[12] Michael Grieves. 2015. *Digital Twin: Manufacturing Excellence through Virtual Factory Replication*. Retrieved June 23, 2024 from https://www.researchgate.net/publication/275211047_Digital_Twin_Manufacturing_Excellence_through_Virtual_Factory_Replication

[13] gRPC. 2024. gRPC - A high performance, open source universal RPC framework. Retrieved June 23, 2024 from https://grpc.io/

[14] HP. 2023. HP Managed Device Services. *HP Managed Device Services*. Retrieved March 3, 2024 from https://www.hp.com/us-en/services/workforce-solutions/workforce-computing/managed-device-services.html

[15] Carlo Human, Anton H Basson, and Karel Kruger. 2023. A design framework for a system of digital twins and services. *Computers in Industry* 144, (January 2023), 103796. https://doi.org/10.1016/j.compind.2022.103796

[16] IBM. 2023. Credential mapping overview. *Credential mapping overview*. Retrieved May 30, 2024 from https://www.ibm.com/docs/en/iis/11.5?topic=configuration-credential-mapping-overview

[17] Intel. 2024. Device as a Service (DaaS): Managing the PC Lifecycle. Retrieved June 28, 2024 from https://www.intel.com/content/www/us/en/business/enterprise-computers/resources/daas.html

[18] ISO/IEC. 2011. *Systems and software Quality Requirements and Evaluation (SQuaRE) System and software quality models*. British Standards Institution. Retrieved August 18, 2023 from https://www.iso.org/standard/35733.html

[19] ISO/IEC. 2018. *Internet of Things - Reference architecture*. British Standards Institution. Retrieved August 18, 2023 from https://www.iso.org/standard/65695.html

[20] Farhana Javed, Muhamamd Khalil Afzal, Muhammad Sharif, and Byung-Seo Kim. 2018. Internet of Things (IoT) Operating Systems Support, Networking Technologies, Applications, and Challenges: A Comparative Review. *IEEE Communications Surveys & Tutorials* 20, 3 (2018), 2062–2100. https://doi.org/10.1109/COMST.2018.2817685

[21] Werner Kritzinger, Matthias Karner, Georg Traar, Jan Henjes, and Wilfried Sihn. 2018. Digital Twin in manufacturing: A categorical literature review and classification. *IFAC-PapersOnLine* 51, 11 (September 2018), 1016–1022. https://doi.org/10.1016/j.ifacol.2018.08.474

[22] Lenovo. 2019. Device as a Service and the circular economy. Retrieved March 3, 2024 from https://static.lenovo.com/ww/docs/ww-lenovo-daas-whitepaper-circular-economy-Dec2019.pdf

[23] Jean-Marc J Maree, Karel Kruger, and Anton H Basson. 2024. Opportunities for Digital Twins for the Provisioning, Management and Monitoring of Heterogeneous IoT Devices. In *International Conference on Engineering Digital Twins*, September 23, 2024. Linz, Austria.

[24] Roberto Minerva, Gyu Myoung Lee, and Noel Crespi. 2020. Digital Twin in the IoT Context: A Survey on Technical Features, Scenarios, and Architectural Models. *Proc. IEEE* 108, 10 (June 2020), 1785–1824. https://doi.org/10.1109/JPROC.2020.2998530

[25] MQTT.org. 2022. MQTT: The Standard for IoT Messaging. *MQTT: The Standard for IoT Messaging*. Retrieved June 23, 2024 from https://mqtt.org/

[26] OCF. 2023. *OCF Security Specification*. Open Connectivity Foundation. Retrieved January 30, 2023 from https://openconnectivity.org/developer/specifications/

[27] Oracle. 2012. *Fusion Middleware Developing Security Providers for Oracle WebLogic Server*. Retrieved May 30, 2024 from https://docs.oracle.com/cd/E24329_01/web.1211/e24486/credmap.htm#DEVSP465

[28] Leila Fatmasari Rahman, Tanir Ozcelebi, and Johan Lukkien. 2018. Understanding IoT Systems: A Life Cycle Approach. *Procedia Computer Science* 130, (April 2018), 1057–1062. https://doi.org/10.1016/j.procs.2018.04.148

[29] Anro Redelinghuys, Anton Basson, and Karel Kruger. 2018. A Six-Layer Digital Twin Architecture for a Manufacturing Cell. In *Service Orientation in Holonic and Multi-Agent Manufacturing* (*Studies in Computational Intelligence*), December 12, 2018. Springer International Publishing, Cham, 412–423. https://doi.org/10.1007/978-3-030-03003-2_32

[30] Jibran Saleem, Mohammad Hammoudeh, Umar Raza, Bamidele Adebisi, and Ruth Ande. 2018. IoT standardisation: challenges, perspectives and solution. In *Proceedings of the 2nd International Conference on Future Networks and Distributed Systems* (*ICFNDS '18*), June 26, 2018. Association for Computing Machinery, Amman, Jordan, 1–9. https://doi.org/10.1145/3231053.3231103

[31] Soraya Sinche, Duarte Raposo, Ngombo Armando, Andre Rodrigues, Fernando Boavida, Vasco Pereira, and Jorge Sa Silva. 2020. A Survey of IoT Management Protocols and Frameworks. *IEEE Communications Surveys & Tutorials* 22, 2 (2020), 1168–1190. https://doi.org/10.1109/COMST.2019.2943087

[32] Maulshree Singh, Evert Fuenmayor, Eoin Hinchy, Yuansong Qiao, Niall Murray, and Declan Devine. 2021. Digital Twin: Origin to Future. *ASI* 4, 2 (May 2021), 36. https://doi.org/10.3390/asi4020036

[33] Susan Symington, William Polk, and Murugiah Souppaya. 2020. Trusted Internet of Things (IoT) Device Network-Layer Onboarding and Lifecycle Management (Draft). https://doi.org/10.6028/NIST.CSWP.09082020-draft

[34] Paul Valckenaers and Hendrik Van Brussel. 2015. *Design for the unexpected: from holonic manufacturing systems towards a humane mechatronics society*. Butterworth-Heinemann. Retrieved from https://doi.org/10.1016/C2014-0-04226-8

[35] Arno H Van Bruggen, Karel Kruger, Anton H Basson, and J Grobler. 2024. An Architecture to Integrate Digital Twins and Machine Learning Operations. In *Service Oriented, Holonic and Multi-Agent Manufacturing Systems for Industry of the Future*, February 03, 2024. Springer Nature Switzerland, Cham, 3–14. https://doi.org/10.1007/978-3-031-53445-4_1

[36] Eric VanDerHorn and Sankaran Mahadevan. 2021. Digital Twin: Generalization, characterization and implementation. *Decision Support Systems* 145, (June 2021), 113524. https://doi.org/10.1016/j.dss.2021.113524

[37] David Walters, Michael Schy, Ashok Bhaskar, Tim Mattison, Alok Jha, and Ben Cooke. 2022. *Device Manufacturing and Provisioning with X.509 Certificates in AWS IoT Core*. AWS. Retrieved June 28, 2024 from https://docs.aws.amazon.com/whitepapers/latest/device-manufacturing-provisioning/device-manufacturing-provisioning.html

[38] Narges Yousefnezhad, Avleen Malhi, and Kary Främling. 2020. Security in product lifecycle of IoT devices: A survey. *Journal of Network and Computer Applications* 171, (December 2020), 102779. https://doi.org/10.1016/j.jnca.2020.102779