

Cloud-Based Battery Digital Twin Middleware Using Model-Based Development

Lukas Merkle

Institute of Automotive Technology
Technical University of Munich, Germany
lukas.merkle@tum.de

Abstract

Following the trends of electrification, the energy storage of vehicles is gaining importance as the most expensive part of an electric car. Since lithium-ion batteries are perishable goods and underlie e. g. aging effects, environmental and operating conditions during manufacturing and car usage need close supervision. With regard to the paradigm of digital twins, data from various life cycle phases needs to be collected and processed to improve the general quality of the system. To achieve this complex task, a suitable framework is needed in order to operate the fleet of digital twins during manufacturing processes, the automotive usage and a potential second life. Based on a literature review, we formulate requirements for a digital twin framework in the field of battery systems. We propose a framework to develop and operate a fleet of digital twins during all life cycle phases. Results feature a case study in which we implement the stated framework in a cloud-computing environment using early stages of battery system production as test bed. With the help of a self-discharge model of li-ion cells, the system can estimate the SOC of battery modules and provide this information to the arrival testing procedures.

CCS Concepts

• **Computer systems organization** → **Cloud computing**; • **Hardware** → **Batteries**; • **Information systems** → *Information systems applications*; *Data analytics*; • **Applied computing** → *Command and control*; *Engineering*.

Keywords

Battery System; Digital Twin; Self-Discharge; Control Middleware; IoT

1 Introduction

The technology of digital twins and cyber physical systems (CPS) is widely used in manufacturing [11]. It is also gaining importance in automotive engineering nowadays. Various components of the car work in a CPS [9] and can partly be modeled as digital twins. One

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ISCSIC 2019, September 25–27, 2019, Amsterdam, Netherlands

© 2019 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-7661-7/19/09...\$15.00
<https://doi.org/10.1145/3386164.3387296>

example is the battery system of electric cars. This most expensive component of the car undergoes close supervision to ensure a safe and long-lasting operation. Following the trend of digital twins, one has to start at the beginning of the life-cycle of a battery system and collect specific data, important for further modeling and prediction of the life of the system. To cope with the different manufacturers as cell producer, battery system assemblers and car manufacturers, but as well as with the usage of the car and a potential second life, a holistic life cycle approach for this digital twin is needed. **This paper proposes a framework which enables the development and operation of a cloud-based digital twin for electric battery systems.** The remainder of this paper is structured as follows: Section 2 gives an overview and comparison of approaches containing digital twin middleware in literature. Subsequently, Section 3 defines requirements and proposes a methodology for a model-based development of digital twins in the field of battery systems. An implementation of the mentioned methodology is outlined in Section 4.

2 Related Work

A digital twin is operated in a computing-environment in parallel with the physical object. At first, it is important to provision appropriate structures for incoming data to save it in standardized ways and to be able to retrieve it for future use. In a preliminary contribution [10], the authors described a basic set of data types for usage in the domain of digital twins for battery systems. A unified modeling language (UML)-metamodel describing the data and information structure of a battery system and its manufacturing process including logistic processes was set up. The result enables concrete modeling of data structures for a digital twin of a battery system. **The methodology presented in this paper builds on top of the prior results.**

2.1 Concepts of Digital Twin Frameworks

This section gives an overview of related approaches defining digital twin architectures including middleware concepts, in various fields of application. Following [7], the concept of *middleware* hides low-level complexity from the application of software and thus allows for clearly defined interfaces.

Schroeder et al. [15] describe a data-modeling methodology in the digital twin context in a production setting. They use AutomationML to model the components of the digital twin, mentioning the ability of object-oriented work flows and hierarchical setup. As base-elements, they use the CAEX meta-model and by that a hierarchy of components while classes define the components including role definitions and attributes. A three-step approach is described, consisting of the modeling process, the deployment of the final model in an

open text-format and, lastly, the usage of the given data structure in various applications. In a case study, they apply their methodology to an industrial valve with the aim of examining the data exchange in digital twin systems.

Schluse and Rossmann [14] describe virtual test beds as an ideal environment for setting up digital twins. In order to build digital twins and overcome limitations in conventional simulation environments, they propose a “Versatile Simulation Database,” which delivers functionality for meta-information management, communication, persistence and user interaction. Liu et al. [8] develop a cloud-based digital twin system for manufacturing machines. Holding master data and bundling in- and outbound data streams, the “Knowledge Ressource Center” (KRC) is the core-element. Connectivity is given through the xml-based MTConnect protocol. By parsing and guiding the inbound data, the KRC sets up connections between physical components and their digital twins.

Yun et al. [16] present a digital twin platform driven by a data centric middleware called uDiT. For their system, they postulate requirements for the data-distributing middleware: it is necessary to connect different twin nodes with each other and synchronize the time between them to generate consistent simulation results. Further on, it is necessary to allow real-time behavior and process multimedia data from the real object into the twin-databases. The uDiT system adds a physical and a co-simulation interface on top of a data distribution service (DDS) [13] stack to serve various simulations and interconnect data from physical objects into the system.

Alam et al. [1] describe more general concepts of communication and data exchange in digital twin environments. In a showcase of an internet of vehicles, a central management authority manages the interconnection of physical and virtual things by starting independent processes for each pair. After establishing the connection, real and digital counterparts can communicate directly via network interfaces. A central service manager carries out governance tasks to control visibility of twin elements and data privacy.

With their MAYA Platform for CPS in production environment, Ciavotta et al. [3] describe a framework for managing and controlling a CPS fleet. The MAYA platform consists of a communication layer, a so called support infrastructure layer and a simulation framework. The communication layer is mainly responsible for orchestration and communication processes. Via the support infrastructure, life cycle phases like birth, data enrichment and dismissal are managed. To access CPS and respective data, it is possible to create communication channels and allow querying against a set of digital twins. Ding et al. [6] describe the shift towards digital twin-driven manufacturing. As a basis for further concepts, they propose a framework using the digital twin and CPS paradigms to control a smart workshop allowing online optimization on shop floor level. Fundamental matters arise in data transfer between master data, e. g. a bill of materials, the cyber- and the physical object.

Bao et al. [2] present a comprehensive work dealing with the modeling and operation of digital twins in the manufacturing processes. They start by formulating methods to build product and process twins, which incorporate an interface, computing and control attributes. Operational digital twins serve as a connection between product and process twins and can help to plan e. g. the task execution sequence or resource allocation alongside the production line.

The Arrowhead framework [4], originating from an EU research project, provides a local clouds-based concept to orchestrate a system of systems in the production field. Core elements addressed in the research project are real-time data handling, security of data and systems, engineering of automation systems and a scaleable, service oriented architecture in a manufacturing context. Following a Service-oriented-Architecture (SoA) paradigm, there are producers and consumers in a system, connected by services. This service-system is controlled and managed by arrowhead-core-systems, where among others a service registry, an orchestration system and an authorization system exist.

2.2 Shortcomings

Section 2 summarizes the aforementioned digital twin middleware concepts. In Table 1, the scopes of the examined approaches are shown, where no contribution delivers a holistic approach regarding the given categories. Under the term governance, we understand functionalities regarding security and access-management for digital twin fleets. Inter-twin communication allows twins to exchange data and thereby allows for a system of multiple digital twin connected systems. Physical / cyber communication relates to any possibility to handle data streams connecting the real and the cyber world. Since that communication is crucial for the digital twin paradigm, all of the mentioned works incorporate this capability. By digitally following a product over a lifetime, life cycle management deals with the birth and virtual assembly of digital twins and the possible hand-over between stakeholders alongside the life cycle. Many publications only deal with manufacturing processes where the issue of transferring virtual objects between various parties is not relevant. In the automotive context with its interlinked value chains, transactions between partners are mandatory. Needing an instance to control single twins as well as a whole fleet, the control instance can either be decentral and by that distributed with the twins or implemented in a central manner. Not all contributions deal with on the control of a digital twin fleet. Applying digital twin technology in the field of battery systems for electric vehicles, a huge and complex system arises. Therefore, a model-based development approach is preferable. Many contributions present results of digital twin frameworks, oftentimes dealing with narrow scopes. The solutions are non-agnostic about the underlying technical infrastructure as well as the data transmitting protocols and simulation frameworks. Not all reviewed contributions arrange for communication inbetween components of the digital twins. The manifold solutions are targeting fields of manufacturing and do not cover other life cycle segments of the product.

None of the mentioned papers examines special requirements for automotive use in the application of battery systems.

The next chapter presents the requirements and a methodology to build a middleware for digital twins of a automotive battery system.

3 Methodology

This chapter reveals requirements for a digital twin middleware, based on a literature review. We Subsequently describe the methodology of our proposed middleware in detail.

Table 1: Comparing features of different digital twin middleware concepts observed in literature.

● given | ⊙ partwise | ○ missing | c.: central | d.: distributed | n.c.: no control

Feature	Schroeder	Yun	Alam	Ciavotta	Ding	Schluse et. al	Liu	Arrowhead	Bao
Governance	○	○	●	●	○	●	●	●	⊙
Inter twin communication	●	●	●	●	●	●	○	●	●
Phys. /cyber communication	●	●	●	●	●	●	●	●	●
Life cycle management	○	○	●	●	●	○	⊙	●	○
Control	n.c.	c.	c.	c.	c./d.	n.a.	n.a.	c.	d.
Model based development	●	○	○	○	○	○	○	○	●
Base technology agnostic	●	●	⊙	●	○	○	⊙	●	⊙

3.1 Identified Requirements

Automotive battery systems as perishable goods need close supervision during manufacturing processes and the life cycle in the car. To embrace the entire lifespan of the battery system and its components, it is necessary to begin the life of the digital twin during early stages of the value added chain. To handle complexity, a modular and model-based approach is preferable. The digital twin architecture needs to be integrated with manifold manufacturers alongside the value-added chain to finally be used in automobiles and in a second life e.g. as energy storage. To support the different manufacturing systems, protocols and environments over the life-time (e.g. manufacturing shop floor vs. usage of automobile), the digital twin architecture needs to be agnostic about the underlying technical infrastructure, respectively highly adaptable to various IT and IoT systems, protocols and simulation environments. With all these requirements, **it is essential to implement the twin architecture in a cloud environment** to use the scaling ability and governance roles provided by the cloud vendor. **Also the software running on top of the cloud vendors stack needs to be highly parallelized and scaleable to keep up with the real world objects.** By relying on web technologies like REST-application programming interfaces (API), easy interchange between different stakeholders becomes possible. Also standard governance tasks like user management and security can be taken from the cloud vendors' offering.

The information flow to run a digital twin is directed from the physical into the cyber world - but not only that direction needs to be intended. Communication between cyber objects is also crucial when it comes to novel services and applications. An example can be the integration of the battery system of a car with a bigger network of things e.g. in smart grid applications. To allow for the communication, a global access interface to the databases including smart ways to access the underlying data needs to be present. To control a single digital twin as well as a fleet of digital twins, it is necessary to implement a control system. Besides life cycle management, e.g. birth of digital twins, the control system can execute certain actions predefined in the digital twins, e.g. polling for data.

3.2 Overview of the Approach

Figure 1 gives an overview of the proposed approach. The scheme is divided into two consecutive parts: the design phase and the run phase of the system. A standardized framework, highlighted in gray, is invariant during the design and run phase and provides features for development of the digital twin fleet as well as for the operation.

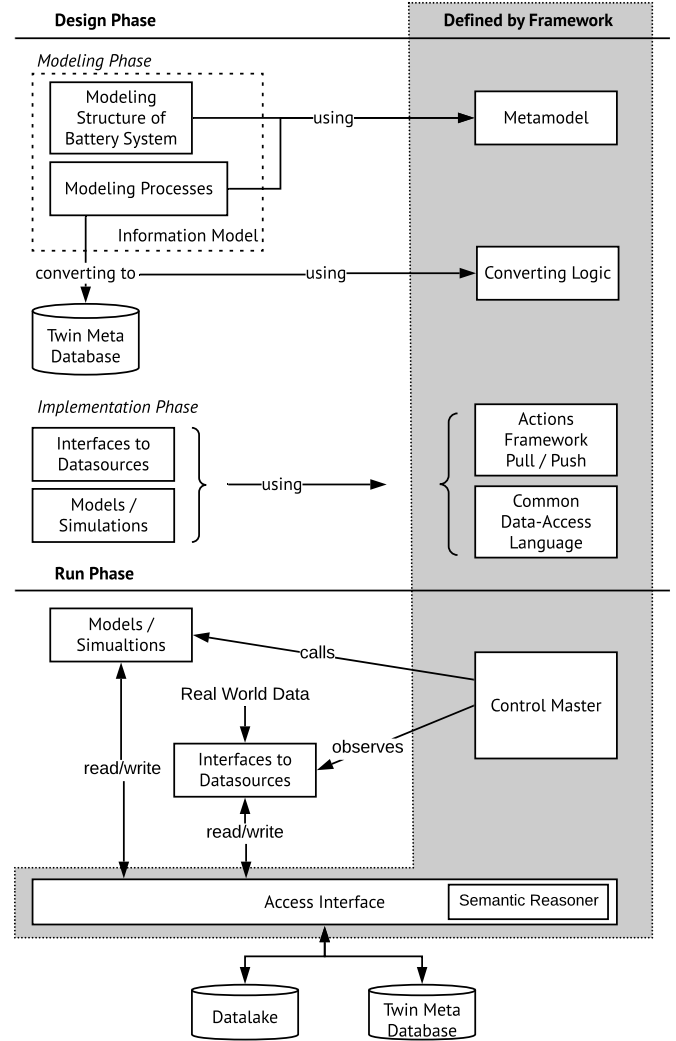


Figure 1: Overview over the phases from digital twin design to the operational. Components that are defined by the framework are highlighted in gray.

3.2.1 Design Phase

We consider the design phase with all aspects related to creating a digital twin of a given real-world thing. Applying the design phase,

it would be integrated in a classical product development workflow as the V-model or the systems development life cycle.

Modeling Phase Resulting from prior work, the UML meta model serves as the starting point for the development process. A first step models the physical object and its processes by relying on the frameworks meta model using its classes and dependencies. Having the system defined, the interface structure of resources like analytic models, simulations and data sources are added into the model. By integrating data interfaces at model-time into the data structure, efforts for a later harmonization of the interfaces can be minimized. Actions describe things that can happen in the running system. Key functionalities of the action-class are to determine parameters and input data as well as handle the resulting data and provide a schedule defining temporal or consecutive calls to the action.

To cope with different kinds of data sources, two main actions “pull action” and “push action” are used. Pull actions are queries against a given endpoint answered in a specific way to the query. The control system executes pull actions based on the information given in a schedule. Possible schedules can be defined relative e. g. execute action five times per hour, or absolute e. g. at a given point in time. Push actions initiate data transfer by themselves by pushing specific data sets to the digital twin architecture.

A sub type of action controls the execution of models or simulations. It is possible to chain actions by defining consecutive actions to orchestrate the system. An example would be the execution of a specific model, once a certain measurement message is received. Having the information model and data sources defined, the model needs to be converted into a suitable database description. By using the frameworks converting logic, a database scheme can be deduced from the UML-Model.

Conversion: UML-Model to NoSQL-Database The data structure as well as the actions from the UML-model need to be interpreted during the conversion process to form a JSON-description of the information model. Using a NoSQL-database that works on a file basis and can directly handle JSON-descriptions we can store the information model. This sets the structure of the digital twin meta database. By not using a fixed table-scheme like in classic relational databases, **NoSQL-databases can be extended easily**. This allows the integration of data structures over the lifetime of the system. During the conversion process, model elements and instance elements as commonly known in object-oriented workflows are distinguished. UML-associations and UML-generalizations are handled as is well-known from the UML specification. UML-datatypes serve as the lowest level in the database hierarchy and are addressed to actually hold the data value. Instance elements like UML-objects, UML-slots and UML-links are used to represent real objects and processes. The objects are thereby defined by UML-classes. UML-links are used to model dependencies between instances. The tool to convert a staruml-model into a json-description can be found here: <https://github.com/TUMFTM/Staruml2json>

Implementation Phase To fill data structures with real world data, the interfaces and models mirroring their physical counterparts need to be implemented, based on the modeling phases specification. This individual implementation of elements can be done with arbitrary,

use-case-matching technology, as long as it integrates with common web interfaces as REST-APIs. It is conceivable that several standard-interfaces, e. g. MQTT [12], can be held available by the framework itself.

To integrate the modules with the control system of the digital twin framework, it is necessary to create defined handler functions, which in turn can be called by the control master.

Standardized Syntax Within the frameworks scope, a uniform syntax is used to specify data paths analogous to UNIX paths. There is a distinction between absolute paths and relative paths. Relative paths are always within the scope of one digital twin. Absolute paths start with the identifier of a specific twin. The semantic reasoner module inside the access interface is also accessible by a respective path.

3.2.2 Run Phase

We consider the run phase with all aspects for operating the fleet of digital twins.

Access Interface Having a standardized way to access all digital-twin-related data is one of the main benefits of the proposed architecture. To achieve this goal, all communication regarding the digital twin databases needs to use the *access interface*. This interface provides methods and classes to mirror the digital-twin database into the current code and is supplying an object oriented workflow. All communication is using the aforementioned syntax.

To make use of domain specific knowledge, the *access interface* features a simple semantic reasoning engine. Domain specific knowledge can be stored in chained rules which the reasoner evaluates. Each rule consists of an antecedent, a subsequent and a transformation from the antecedent to the subsequent. To get predictable results if two subsequents hold the same information which they generate from different antecedents, each rule holds a priority. The lowest priority value is favored.

Control Master The control master describes a central instance for controlling in- and outbound data streams as well as managing calls to models of the digital twin objects in a temporal and sequential manner. Following a client-server architecture, diverse handlers for sending and gathering data as well as for executing models can register with the control master. Using a central master instead of a distributed control scheme allows other components, e. g. models, to remain focused on their task without having to deal with data in- and output operations.

The control master reads the control information (so-called actions) from the twin meta data base specifically for every single digital twin. Actions define e. g. which model is to be executed after receiving a specific dataset or poll sensors in a regular schedule. Actions also determine where to get information needed for their execution and where to put eventual results.

To execute actions, each type of action needs an action handler registered with the control master. The action handlers are executable scripts interpreting the specific action. As described in the modeling phase, actions can be of different types. Depending on the schedule and trigger information saved in a pull action, the control master executes the denoted handler which in turn executes the action and handles associated in- and outbound datastreams. Any data needed to execute the action is either gathered from the digital twins

databases using the access interface or gathered from an external source, e.g. a weather API. To allow for the scaling of this architecture, handlers are started as individual processes. To handle push actions, the control master needs to listen to all channels providing push actions to the twin database. If one channel reports a message, the control master interprets the message and directs its content to the corresponding twin element. Listeners for push channels run in separate processes to allow concurrent reaction and scaling. Receiving push data, the system needs to be able to extend the data structure by adding new structural items, or even new digital twins if none of the data-pushing physical items is set up already. Actions can define consecutive actions. The control master takes care to call them once the preceding action is returning. Orchestration tasks such as calling a model if a specific dataset is updated, can be achieved using the consecutive actions functionality. If new twins are created and saved in the database, the birth handler reports this back to the control master. To register the actions of the new twin, the control master is triggered to read in the database and update the internal list of actions.

4 Results and Case Study

With the help of the proposed architecture, a digital twin of battery modules was developed, implemented and tested in a case study. The environment is defined by the early steps of battery system manufacturing shown in Figure 2. The goal of the architecture is an estimate for the self-discharge of the transported battery cells. In an arrival step at the battery systems manufacturer, cells undergo testing to ensure that their open circuit voltage (OCV) is within a specified window of ± 10 mV. Taking self-discharge into account, one could implement variable thresholds depending on the predicted self-discharge due to transport time and temperature. Temperature, moisture and acceleration during the shipping of the battery modules within Europe were recorded using a data logger. In the future, smart carriers deliver the data via IoT technology; however, for this study, we developed a simulator replaying the measured data into an IoT stream.

Technical Testbed The case study was carried out using Amazon Web Services (AWS) as the cloud provider. Figure 2 depicts the cloud vendor level as the second level above the real world. In Table 2, the technologies utilized are listed. We use storage services like AWS Simple Storage (S3) and DynamoDB to store real-world data and twin meta data. Building a vendor-specific access layer (vsal), computing services like the Lambda service are interfacing with the persistence layer. Vsal can be addressed via REST-API calls by the digital twin access interface. This makes the digital twin access interface agnostic to the vendor of the cloud service. A model of the self-discharge of li-ion cells is run via the AWS Fargate Service that is capable of running docker containers.

4.1 Design Phase - Setting up the Information Model

Using the meta model for digital twins of battery systems from [10], a UML model of the battery modules and their logistics processes

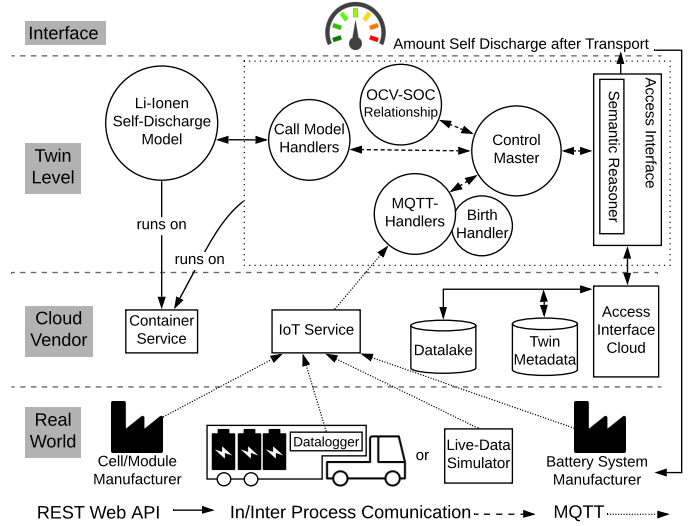











Figure 2: Environmental Testbed and virtual levels of the Case Study.

Table 2: Technologies used for the elements in the digital twin architecture.

Element	Technology
Cloud Vendor	Amazon Web Services (AWS)
IoT-Protocol	MQTT 
Persistence	AWS S3 
Meta Database	AWS DynamoDB 
VSAL	AWS Lambda (Python) 
Control Master	Python Program 
Access Interface	Python Module 
Self Discharge Model	Implemented in Python, run on AWS Fargate (Dockerized)   

is defined. Data sources like the datalogger in the truck where integrated into the model in this step. Further on, we can integrate control information into the model, e.g. an action that is calling the self-discharge model once a new temperature measurement reaches the database. Having a complete information model defined in UML, the information model is converted into a json description, which forms the basis of the twin meta database.

4.2 Design Phase - Implementing the System

4.2.1 Live-Data Simulator

In the case study, a simulator is used to replay the collected data from the transport. Implemented in python, the simulator sends MQTT messages according to the underlying raw data. All steps from birth, storage, transport over testing at the cell manufacturer to testing at the battery systems manufacturer are represented by specific messages from the simulator.

4.2.2 Access Interface

The access interface to the digital twin data is implemented in python and acts as an abstraction layer between the cloud provider and the digital twin applications. Classes encapsulate the database access and allow an easy integration into the code for e. g. interfaces of the control master. By using the ontology-based semantic reasoner, the access interface can reason about the best matching answer to a request. As an example, one can ask for the temperature trend during a specific situation e. g. in a specific storage. By evaluating rules, the access interface returns with a data logger measured temperature trend if available. Otherwise, it returns a “best guess”, i. e. a temperature trend generated from the temperature set point of the storage over the stored time.

4.2.3 Control Master

The control master acts as a central management instance to operate the fleet of digital twins. In the case study, several handlers are implemented in the control master. MQTT handlers are listening for incoming push messages and accordingly save the payload to the database. The self-discharge model as well as the open circuit voltage (OCV)-state of charge (SOC) relationship are called from own handlers. As a general tasks, the control master generates new entries in the digital twin meta-database, if unknown, hence new battery modules are observed. This task is handled by the birth handler. The control master can either be run on local machines for testing purposes or be deployed into the cloud environment via docker containers. All elements of the control master are implemented in python.

4.2.4 Self-Discharge Model

For the case study, we use a self-discharge model for li-ionen cells based on the work of Deutschen et al. [5]. In our application, the model for short time periods is implemented using python and run in a docker container. We use the parameter sets given by the authors in [5] and interpolate between paramter sets for 25 °C and 60 °C for the temperature input. Input data is posted to the model via REST Web-API in json-notation. The model returns the estimated SOC value including self-discharge over the given time period depending on the given temperature. The model is plausibilized using measurements of our testbed battery module in one point of temperature. The model serves as a black box to test the overall system and is not validated over the whole temperature range, which needs to be done prior to productive use.

4.3 Run Phase

The system was tested tracking one charge of 181 battery modules delivered from the cell and module manufacturer to the system manufacturer. We use the IoT simulator to send the messages. To initialize the system, a birth message is sent for each module to initialize the process. Based on serial numbers in the messages, a digital twin object with the respective structure is created in the database and its handlers in the control master are started. After initializing, the battery module manufacturer sends the time and the results of the end of line (EOL) test of the module into the digital twin database. The EOL test includes a measurement for the OCV of the module which the control master, using the respective handler, converts into

a SOC using the OCV-SOC relationship in the digital twins data set. The calculated SOC is the starting point for the self-discharge model tracking from now on and is called in a regular fashion. During transport, data regarding temperature and GPS position is sent to the database. Arriving at the systems manufacturer, the modules are stored in a temperature-controlled environment before they are tested in an arrival test and integrated into the battery system. Information about the start and end of storing as well as the measurement data of the arrival test are sent into the twin database. Using the semantic reasoner, the self-discharge model is fed with the best available temperature history since the start of life and returns an estimate for the current self-discharge of the whole module and the single cells. The value can be obtained for continued usage from the graphical user interface (GUI).

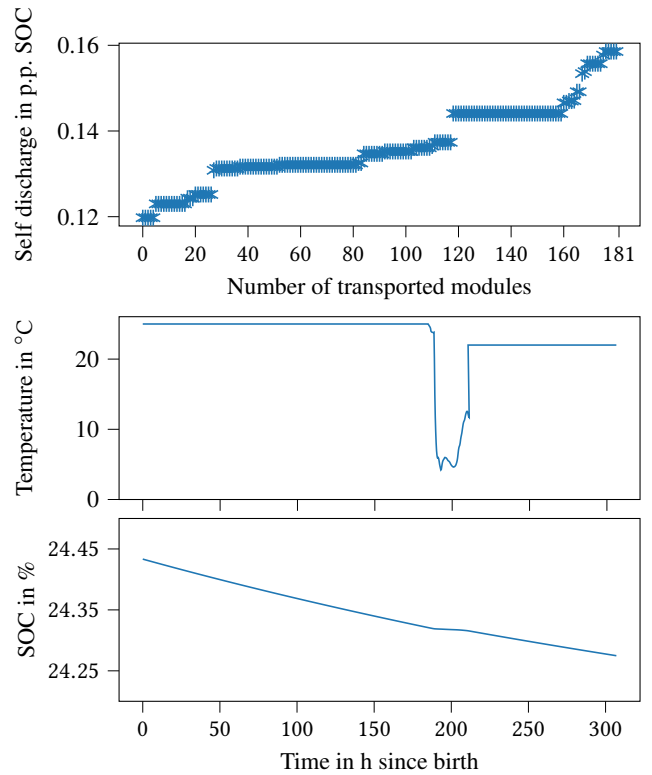


Figure 3: (a) Modeled self-discharge of all modules sorted by age. (b) One exemplary temperature trend obtained from the reasoning module. (c) Modeled trend of self-discharge over the given temperature profile.

In Figure 3(a), we show the modeled self-discharge sorted by the age of the modules. Older modules are up to 300 hours old, while the youngest modules were stored less time between production and are only 250 hours old. A longer life results in longer storage time and by that higher self-discharge. In Figure 3(b), we show one exemplary temperature trend of one module obtained from the systems semantic reasoner. The period at 25 °C denotes to the storage at the module manufacturer. During the transportation, temperatures

where logged and declining to 5 °C before the temperature is rising to 22 °C at the storage of the battery systems manufacturer. The steady periods at 25 °C and 22 °C were not logged but are a best guess, based on the temperature set point in the definition of the storage environment.

In Figure 3(c), we show one exemplary self-discharge trend calculated by the model. Having an estimate of the self-discharge of the battery cells and modules according to the storing and transportation conditions, one could take that into account during the arrival test, where one needs to decide if one module can be used for further production or if it needs to be rejected. Thresholds for distinguishing good from bad modules can be set up dynamically onto the particular history of each examined module.

5 Conclusion

In this work, we present a novel approach for engineering and running a fleet of digital twins in the domain of battery systems. By the model-based engineering approach, complexity can be handled. The central element of the approach is the multi-level architecture which uses a cloud-infrastructure for computing and communication. In the case study, a potential implementation is shown, which indicates the operability of the developed methodology. Further research should be directed towards the control strategies to allow for the most efficient usage of computation power and network utilization. Further on, services running on top of the architecture need to be developed and tailored to the customers' needs.

ACKNOWLEDGEMENTS

Lukas Merkle initiated the idea of the paper and developed the methods and results. Markus Lienkamp gave final approval of the version to be published and agrees to all aspects of the work. As a guarantor, he accepts responsibility for the overall integrity of the paper. We thank the DRÄXLMAIER Group for the cooperation and necessary information provided.

References

- [1] ALAM, K. M., SOPENA, A., AND SADDIK, A. E. Design and Development of a Cloud Based Cyber-Physical Architecture for the Internet-of-Things. *Proceedings - 2015 IEEE International Symposium on Multimedia, ISM 2015* (2016), 459–464.
- [2] BAO, J., GUO, D., LI, J., AND ZHANG, J. The modelling and operations for the digital twin in the context of manufacturing. *Enterprise Information Systems* 0, 0 (2018), 1–23.
- [3] CIAVOTTA, M., ALGE, M., MENATO, S., ROVERE, D., AND PEDRAZZOLI, P. A Microservice-based Middleware for the Digital Factory. *Procedia Manufacturing* 11, June (2017), 931–938.
- [4] DELSING, J. Arrowhead, 2019.
- [5] DEUTSCHEN, T., GASSER, S., SCHALLER, M., AND SIEHR, J. Modeling the self-discharge by voltage decay of a NMC/ graphite lithium-ion cell. *Journal of Energy Storage* 19, June (2018), 113–119.
- [6] DING, K., CHAN, F. T., ZHANG, X., ZHOU, G., AND ZHANG, F. Defining a Digital Twin-based Cyber-Physical Production System for autonomous manufacturing in smart shop floors. *International Journal of Production Research* 0, 0 (2019), 1–20.
- [7] ETZKORN, L. H. *Introduction to Middleware*, 1 ed. Chapman and Hall/CRC, New York, 2017.
- [8] LIU, X. F., AL SUNNY, S. M. N., NGUYEN, N.-T., TAO, W., LEU, M. C., SHAHRIAR, M. R., AND HU, L. Modeling of Cloud-Based Digital Twins for Smart Manufacturing with MT Connect. *Procedia Manufacturing* 26 (2018), 1193–1203.
- [9] LUKASIEWYCZ, M., STEINHORST, S., SAGSTETTER, F., CHANG, W., WASZECKI, P., KAUER, M., AND CHAKRABORTY, S. Cyber-Physical Systems Design for Electric Vehicles. In *2012 15th Euromicro Conference on Digital System Design* (sep 2012), IEEE, pp. 477–484.
- [10] MERKLE, L., SEGURA, A. S., TORBEN GRUMMEL, J., AND LIENKAMP, M. Architecture of a Digital Twin for Enabling Digital Services for Battery Systems. In *2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS)* (may 2019), IEEE, pp. 155–160.
- [11] MONOSTORI, L., KÁDÁR, B., BAUERNHANSL, T., KONDOH, S., KUMARA, S., REINHART, G., SAUER, O., SCHUH, G., SIHN, W., AND UEDA, K. Cyber-physical systems in manufacturing. *CIRP Annals* 65, 2 (2016), 621–641.
- [12] OASIS STANDARD. Mqtt-v3.1.1- Specification, 2014.
- [13] OMG. Data Distribution Service Specification Version 1.4, 2015.
- [14] SCHLUSE, M., AND ROSSMANN, J. From Simulation to Experimentable Digital Twins. *Systems Engineering (ISSE)*, 2016 *IEEE International Symposium on* (2016), 1–6.
- [15] SCHROEDER, G. N., STEINMETZ, C., PEREIRA, C. E., AND ESPINDOLA, D. B. Digital Twin Data Modeling with AutomationML and a Communication Methodology for Data Exchange. *IFAC-PapersOnLine* 49, 30 (2016), 12–17.
- [16] YUN, S., PARK, J. H., AND KIM, W. T. Data-centric middleware based digital twin platform for dependable cyber-physical systems. *International Conference on Ubiquitous and Future Networks, ICUFN* (2017), 922–926.