EUROPEAN COMMISSION

HORIZON EUROPE PROGRAMME – TOPIC: HORIZON-CL5-2022-D2-01

FASTEST

Fast-track hybrid testing platform for the development of battery systems

Deliverable D5.2: Digital Twin models definition

Primary Author [Marco Rodrigues]

Organization [INEGI]

Date: [20.03.2025]

Doc. Version: [V1.0]

Co-funded by the European Union under grant agreement N° 101103755 and by UKRI under grant agreement No. 10078013, respectively. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Climate, Infrastructure and Environment Executive Agency (CINEA). Neither the European Union nor CINEA can be held responsible for them.

Document Control Information	
Settings	**Value**
Work package:	5 – Digital Twin for Hybrid Test in Battery Development
Deliverable:	5.2
Deliverable Type:	R – Document, report
Dissemination Level:	PU - Public
Due Date:	31.03.2025 (Month 22)
Actual Submission Date:	31.03.2025
Pages:	< 40 >
Doc. Version:	V1.1
GA Number:	101103755
Project Coordinator:	Bruno Rodrigues | ABEE (bruno.rodrigues@avestaholding.com)

Formal Reviewers		
Name	**Organization**	**Date**
Mariana Fernandez	SIE	21/03/2025
Eliana Giovannitti	COMAU	27/03/2025

Document History			
Version	**Date**	**Description**	**Author**
V0.1	06/03/2025	First version	Marco Rodrigues (INEGI)
V0.2	15/03/2025	Second version	Marco Rodrigues (INEGI)
V0.3	19/03/2025	Third version	Nuno Marques (INEGI)
V1.0	21/03/2025	Internal quality review	Mariana Fernández (SIE)
V1.1	27/03/2025	Review	Eliana Giovannitti (COMAU)

Project Abstract

Current methods to evaluate Li-ion batteries safety, performance, reliability and lifetime represent a remarkable resource consumption for the overall battery R&D process. The time or number of tests required, the expensive equipment and a generalised trial-error approach are determining factors, together with a lack of understanding of the complex multiscale and multi-physics phenomena in the battery system. Besides, testing facilities are operated locally, meaning that data management is handled directly in the facility, and that experimentation is done on one test bench.

The FASTEST project aims to develop and validate a fast-track testing platform able to deliver a strategy based on Design of Experiments (DoE) and robust testing results, combining multi-scale and multi-physics virtual and physical testing. This will enable an accelerated battery system R&D and more reliable, safer and long-lasting battery system designs. The project's prototype of a fast-track hybrid testing platform aims for a new holistic and interconnected approach. From a global test facility perspective, additional services like smart DoE algorithms, virtualised benches, and Digital Twin (DT) data are incorporated into the daily facility operation to reach a new level of efficiency.

During the project, FASTEST consortium aims to develop up to TRL 6 the platform and its components: the optimal DoE strategies according to three different use cases (automotive, stationary, and off-road); two different cell chemistries, 3b and 4 solid-state (oxide polymer electrolyte); the development of a complete set of physic-based and data-driven models able to substitute physical characterisation experiments; and the overarching DT architecture managing the information flows, and the TRL6 proven and integrated prototype of the hybrid testing platform.

LIST OF ABBREVIATIONS, ACRONYMS AND DEFINITIONS

Acronym	Name
AI	Artificial Intelligence
DT	Digital Twin
DTDL	Digital Twins Definition Language
DoE	Design of Experiments
EC	European Commission
FAIR	Findable, Accessible, Interoperable, and Reusable
IoT	Internet of Things
IT	Information Technology
LIMS	Laboratory Information Management System
MQTT	Message Queuing Telemetry Transport
NASA	National Aeronautics and Space Administration
OWL	Web Ontology Language
T	Task
UUID	Universally Unique Identifier
WP	Work Package
ZENODO	Open repository for EU-funded research outputs from Horizon Europe

Table of Contents

1. Executive summary

This report details the Digital Twin models for the FASTEST project, supported by the European Union’s Horizon Europe Research and Innovation Program under Grant Agreement number 101103755. The primary objective of the Digital Twin models is to provide a reliable representation of the behavior of cells, modules and battery packs in the tests to be virtualized. For that, it outlines the FASTEST Digital Twin Models, which are an output of T5.2, and are a contribution to other future applications and/or other domains in the field of physical systems testing. The FASTEST Models represent structured technology-agnostic entities for Data, Digital Twin and Communication standard design.

FASTEST Digital Twin Models advance beyond state-of-the-art, by leveraging the virtualization for testing cells, modules and battery packs. These developments will enable the scientific community to build Digital Twin applications on top of the results, leveraging the digital transition for the battery universe, boosting future integrations of building blocks based on AI and/or advanced data analysis for process optimization.

The key progress achieved includes the FASTEST Data Models, Digital Twin Models, Communication and Metadata Models for three use-case applications (Stationary, Automotive and Off-Road) regarding battery testing, validated through designed test cases with consortium partners. The developed Models target battery testing and can be adjusted for slightly different domains, providing a structured framework to specify data information systems and communication structures.

2. Objectives

The objective of this report is to provide a structured approach to understanding and implementing Digital Twin Models by defining their core elements, categorization, and interaction mechanisms. By exploring the evolution of Digital Twin Ontology, this report aims to establish a clear distinction between different types of models, including Data Models, Digital Twin Models, Communication Models and Metadata Models, providing these as output of this task. The development of these Models considered three different use cases despite being adaptable and flexible to have more in the future, physical attributes of the assets, testing requirements, and integration requirements with WP2, WP3, WP4 and WP6 as well as the necessary supporting modules identified in WP1 (particularly T1.4).

This study highlights the importance of standardization in Digital Twin development to ensure interoperability, accurate representation and smooth integration with physical and digital environments. Additionally, it aims to define the role of core elements and services in enabling efficient data management, decision-making, and system optimization within Digital Twin ecosystems.

Through the activities developed within this task scope, the report provides a foundational framework for FASTEST Digital Twin and for organizations looking to develop, enhance, or integrate Digital Twins into their operational workflows, facilitating improved efficiency, reliability, and innovation.

3. Introduction

The concept of the Digital Twin has drawn interest from a variety of industrial and engineering fields. A Digital Twin is a virtual version of a physical system that allows for real-time data integration, predictive analysis, and assistance. NASA first used them for spaceship modeling and monitoring (Glaessgen & Stargel). Digital Twins provide problem detection, operational forecasting, and system optimization through the constant flow of information between the digital and physical equivalents (Gupta, Khare, & Rawat, 2023).

Digital Twin models have evolved into essential tools for industrial digital transformation. These models range from simple data-driven representations to highly complex physics-based simulations that integrate real-time sensor data and advanced analytics.

In the context of battery testing, Digital Twin models enable manufacturers and researchers to accelerate product validation by simulating charging cycles, thermal behavior, and degradation patterns. By integrating real-time sensor data with predictive analytics, these models facilitate early detection of defects and support the development of improved battery management systems. This reduces reliance on extensive physical testing and shortens the product development lifecycle. By leveraging Digital Twin technology, battery testing processes can be optimized, ensuring that new battery technologies are not only developed faster but also meet reliability and safety standards. The ability to create a virtual testing environment where multiple scenarios can be simulated enables manufacturers to identify potential issues at an early stage, significantly reducing the need for costly and time-consuming physical prototypes. Moreover, by continuously refining models with real-world data, Digital Twins enhance the accuracy of failure predictions and enable adaptive testing strategies that evolve alongside battery technology advancements. Digital Twins also contribute to a broader digital transformation initiatives by enabling interconnected, data-driven decision-making processes. Through cloud computing, edge analytics, and IoT integration, these models support automation and remote monitoring of battery testing environments, improving efficiency and scalability (Duan, Gao, Yang, & Li, 2022). The ability to monitor battery performance in real time and compare digital predictions with actual test results enhances confidence in predictive models, allowing users to refine their approaches and reduce uncertainties in battery development. However, challenges such as data integrity, computational complexity, and cybersecurity concerns must be addressed to ensure the successful deployment of Digital Twin technologies in battery testing applications.

In the context of FASTEST, Digital Twin Models were developed in parallel with Data Models, Communication Model and Metadata Models, and all these serve as virtual representations that replicate the cells, modules or battery packs behavior under various testing conditions for three different use cases, automotive, stationary and off-road, providing a reliable and technology-agnostic data structure that represents the battery testing domain and serves as the foundation for the Digital Twin Platform, output of WP5, being these also easily adaptable to different physical-system testing domains.

4. Evolving Digital Twin Ontology into Digital Twin Models

Transitioning from an ontology-based framework that resulted as output from T5.1 in deliverable 5.1, to a structured Digital Twin based in Data Models, Digital Twin Models, Communication and Metadata Models requires a methodological approach to ensure that conceptual relationships, data integrity, and interoperability are maintained. This transformation involves several critical steps that align ontology-driven knowledge representation with practical implementation in Digital Twin environments.

To develop a standardized methodology for evolving ontology into a Digital Twin model, the following steps were adopted:

1. Ontology Characterization and Mapping: Identify the key concepts, entities, relationships, and constraints within the ontology that need to be transformed into Digital Twin models. Ensure that domain-specific knowledge is translated into the model structure.
2. Definition of Standardized Data Representation: Establish a structured data representation based on widely accepted standards such as OWL (Web Ontology Language), DTDL (Digital Twins Definition Language), or RDF (Resource Description Framework). Define how hierarchical structures will be mapped into Digital Twin components and Data Entities.
3. Implementation of Digital Twin Components and Data Models: Convert ontology-defined entities into Digital Twin components such as assets, telemetry points, relationships, and event-driven behaviours. Utilize a modular approach where each component maintains well-defined properties and relationships. Use the same approach for Data Modelling based on the Ontology.
4. Interoperability and Integration Strategy: Define communication interfaces to integrate the Digital Twin Models with existing systems, Laboratory Information Management Systems (LIMS) and co-simulation platform. Ensure compatibility with data exchange protocols such as MQTT or RESTful APIs.
5. Validation and Iterative Refinement: Constant monitoring between the Models designed for the Digital Twin and the data representation and refining as new information is acquired, in order to enhance accuracy.
6. Deployment and Continuous Evolution: Once validated, deploy the Digital Twin Models within a cloud or edge computing infrastructure. Establish mechanisms for continuous learning and adaptation, allowing the Digital Twin to evolve based on new data and insights.

By following these structured steps, the transition from ontology to Digital Twin Models ensures that the conceptual framework is preserved while enabling real-world applications in predictive analytics, decision support, and system optimization. This methodology facilitates the development of robust and scalable Digital Twins for industrial applications, particularly in battery testing and related domains.

5. Categorization of the Digital Twin

Digital Twins can be categorized into several types based on their scope, functionality, and level of integration:

- Product Digital Twin: A virtual model of a specific product or component, used throughout its lifecycle (from design to operation).
- Process Digital Twin: A simulation of a process or workflow to optimize efficiency, reduce waste, and enhance decision-making.
- System Digital Twin: A DT that represents multiple interconnected components within a larger system, ensuring system-wide optimization.
- Hybrid Digital Twin: A combination of multiple DT types (Product, Process, Asset, System) to provide comprehensive insights across different domains.

Within this categorization, the specific DT for this project is classified as a Hybrid Digital Twin, as it has the product (battery cell, module or pack) and the process (test).

Depending on the level of integration, there are also three types:

- **Digital Model**: The data between physical and virtual objects are manually updated by the user. Any change to the state of the physical object is not reflected in the virtual object, and any change to the state of the virtual object is not reflected in the physical object.

- **Digital Shadow**: The data of the physical object is automatically transmitted to the virtual object, so there is one-way communication between the two objects.

- **Digital Twin**: Involves a two-way data transmission between the physical twin and the digital one, and any modification made to both the physical and the digital object will be reflected in the behavior of its counterpart.

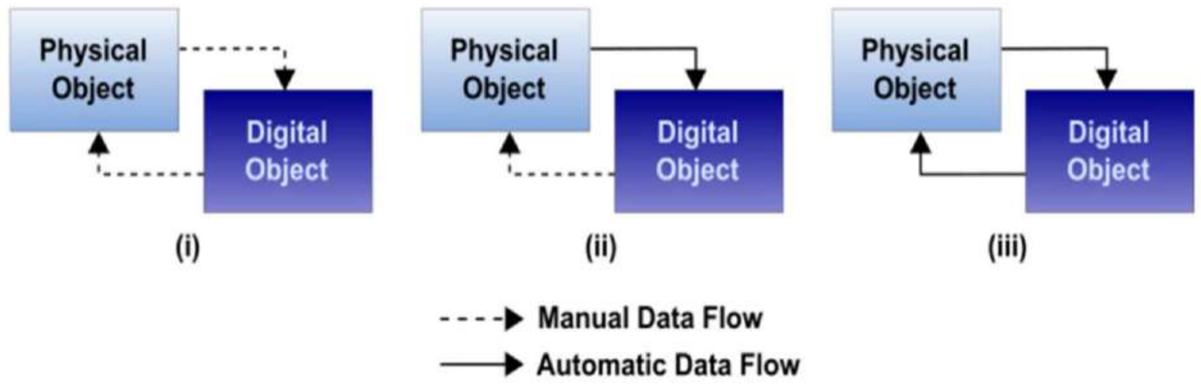


Figure 1 - Types of Digital Twin depending on data exchange

FASTEST WP5 Platform classifies as Digital Twin, since the test results of the WP3 simulations will be made available automatically to the Digital Twin user and will serve as input for the toolchain's data-driven models (WP4). This interoperability between models and internal components will be provided by the Digital Twin.

6. Digital Twin Core Elements and Services

The FASTEST project employs a Digital Twin framework to accelerate battery testing through real-time data integration from physical and virtual test simulations. To effectively implement this system, specific core elements and components have been identified that support the modelling, monitoring, and optimization of battery performance. These elements ensure interoperability with WP6 platform, as well as between Digital Twin's multiple internal core modules, in order to ensure that each one can perform its function, and the system as a whole can virtually and reliably represent the behaviour of battery components in test procedures for the three considered use cases. The integration of these internal Digital Twin modules will make it possible to create 3 instances of Digital Twin, applied to each of the Use Cases, thus achieving Objective 3 of the FASTEST project.

The Digital Twin internal core elements, that were first identified in deliverable D1.3 – Requirements and Specifications for Digital Twins as requirements for the DT, are now comprehensively technically detailed in D5.3 - Integration plan for Digital Twin on the platform are:

- **Model Upload** – interface for uploading, updating, and securely storing various types of simulations and models developed within the scope of WP3 and 4.
- **Test Handler** – internal service to handle all steps in between the tests' execution, except for running the Simulations or Models, starting from receiving an order to execute a specific test, specify which simulations and test procedures for the Unit Under Test should run, etc.

- **Data Management** – collect and store the output data generated by running simulations and physical tests, as well as handle this data and store it in adequate structures for further analysis.
- **Real-Time Monitoring** - deliver real-time information about ongoing simulations via its outputs.
- **Communication Monitoring** - ensure that all Digital Twin resources remain operational overtime.

The implementation of Digital Twin's core modules is based on a set of platform-agnostic models, which will be able to be reused in future Digital Twin application research projects, either directly in the context of battery testing or in slightly different contexts, in which case adaptation may be necessary. Such Models are detailed in the following chapter and will be shared in Zenodo platform for further research initiatives.

7. Models Definition

In the Information Technology (IT) domain, a model is a structured representation of a system, process or concept, which details all the relevant characteristics so that the entity it represents is able to perform all its functions. This way, using Models to represent a system ensures that it is represented in technology-agnostic structures, as they can always be transposed and reused. FASTEST Digital Twin is composed of several models that represent different information structures and define physical characteristics of each asset, operational parameters, and other relevant conditions, which in turn make up different functions of the system as a whole, and serve as the foundation for accurately representing and managing battery testing procedures.

7.1. Data Models

Data Models are a structured framework that define how data is organized, stored, and managed within a system, ensuring consistency, integrity, and accessibility. It establishes the relationships between data elements, the rules governing data interactions, and the formats for storing and retrieving information. FASTEST Data Models aims to depict an organised data structure that represents the domain of cell, module and battery packs testing, for the three considered use-cases, automotive, off-road and stationary, serving as the basis of Digital Twin, allowing data management and maintenance over time, and providing the necessary foundations for the development of the system from an application point of view, which are described in D5.3 Integration plan for Digital Twin on the platform with the specification of the system's architecture and software components.

FASTEST Data Models can be seen in Figure 2, Figure 3, Figure 4, and consist of the following related entities:

1. **Battery Packs**

- **Pack (Characterization)** – Generic and static properties for each Pack. Information that will never change, such as *Pack Capacity*, *Voltage, Number of Modules, Operating Temperature Max, Pack Weight*, among others.
- **Pack Instance** – Small model that allows multiplication of packs for the same characterization.
- **Pack Test Characterization** – Generic properties for Pack Tests. Basic information to represent the test, such as *Test Name, Test bench* or *Simulation Version* and the *Use Case*. This model is connected to the *Pack Test Procedures*, *Pack DoE Test Procedures* , *Pack Test Inputs* and *Pack Test Outputs* models, as each *Test* will need data structures to store this related data.
- **Pack DoE Test Procedures** – Configuration for each Test Procedure as output from the *DoE* for one test. Considers properties such as *Temperature, Voltage* or *Current*.
- **Pack Test Procedures** – Configuration for each standard Test Procedure suggested by the Digital Twin to the *DoE* for one test. Considers properties such as *Temperature, Voltage* or *Current*.
- **Pack Test Inputs** – Represents the inputs for each Pack Test. Considers *Time, Voltage* and *Current*.
- **Pack Test Outputs** – Represents the outputs for each Pack Test. Considers *Time, Temperature* and *Capacity*.

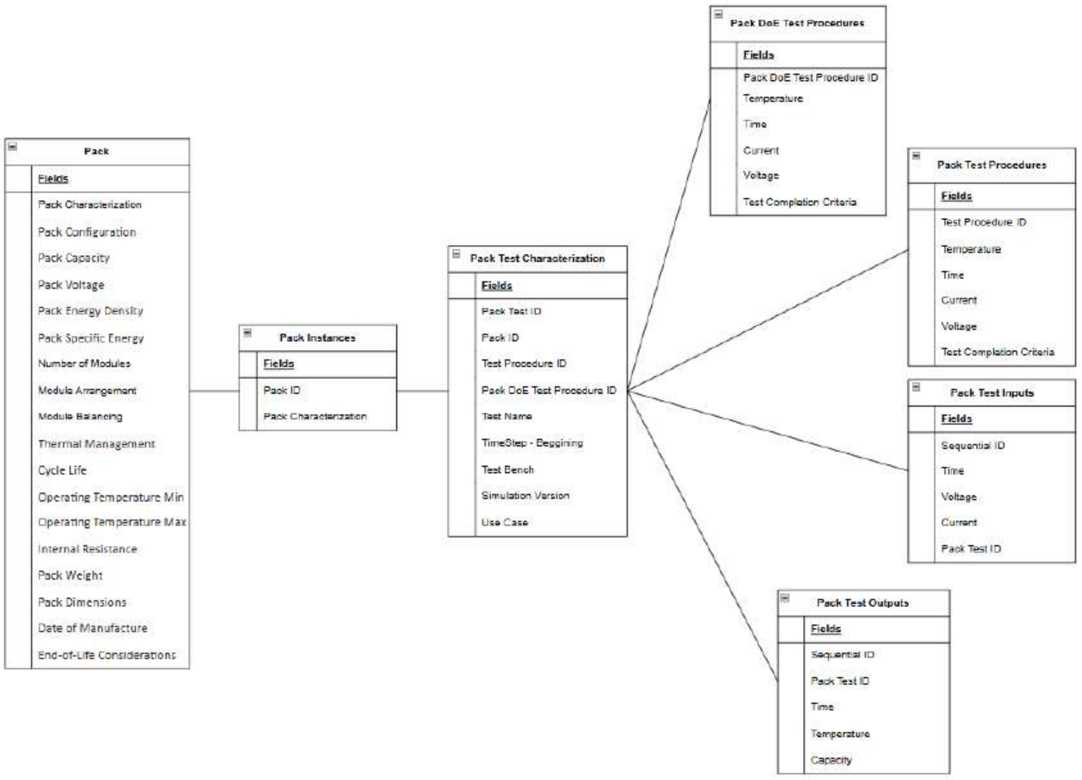


Figure 2 - Data Models schema for Pack Testing

2. <u>Module</u>

- **Module (Characterization)** – Generic and static properties for each module. Information that will never change, such as *Module Capacity, Voltage, Number of Cells, Operating Temperature Max, Module Weight*, among others.
- **Module Instance** – Small model that allows multiplication of modules for the same characterization.
- **Module Test Characterization** – Generic properties for Module Tests. Basic information to represent the test, such as *Test Name, Test bench* or *Simulation Version* and the *Use Case*. This model is connected to the *Module Test Procedures, Module DoE Test Procedures , Module Test Inputs* and *Module Test Outputs* models, as each *Test* will need data structures to store this related data.
- **Module DoE Test Procedures** – Configuration for each Test Procedure as output from the *DoE* for one test. Considers properties such as *Temperature, Voltage* or *Current*.
- **Module Test Procedures** – Configuration for each standard Test Procedure suggested by the Digital Twin to the *DoE* for one test. Considers properties such as *Temperature, Voltage* or *Current*.
- **Module Test Inputs** – Represents the inputs for each Module Test. Considers *Time, Voltage* and *Current*.
- **Module Test Outputs** – Represents the outputs for each Module Test. Considers *Time, Temperature* and *Capacity*.

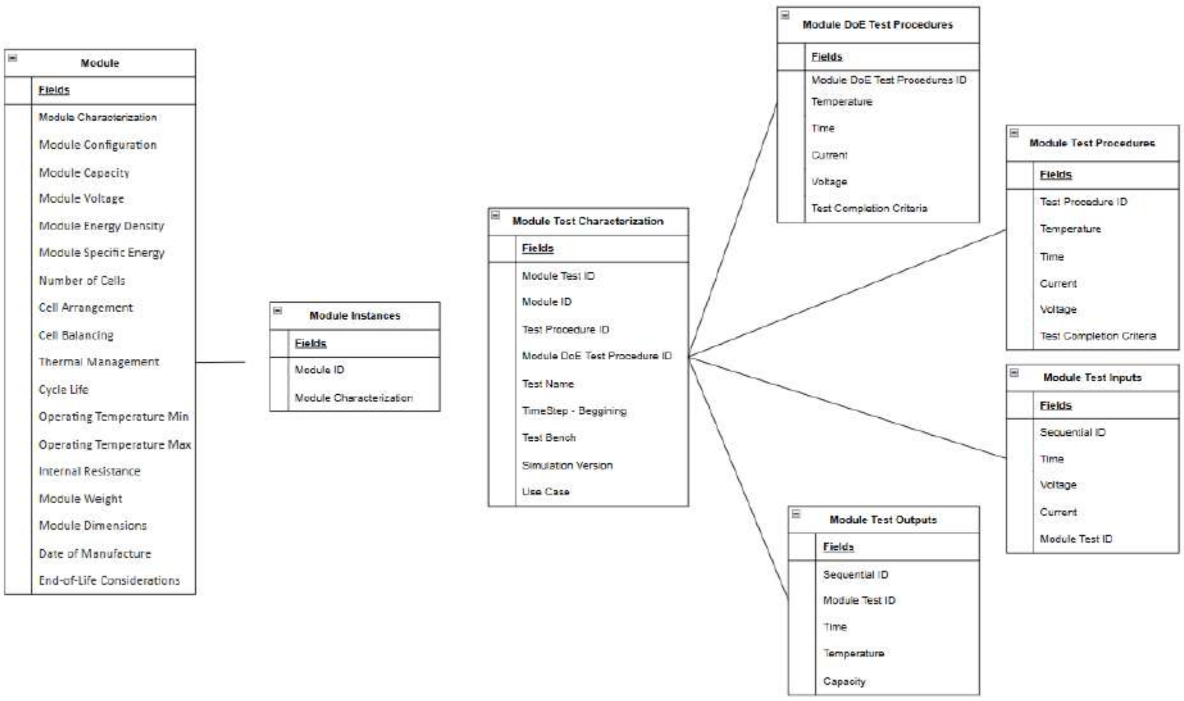


Figure 3 - Data Models schema for Module Testing

3. Cell

- **Cell (Characterization)** – Generic and static properties for each Cell. Information that will never change, such as *Chemistry, Voltage, Energy Density, Operating Temperature Max, Cell Weight*, among others.
- **Cell Instance** – Small model that allows multiplication of cells for the same characterization.
- **Cell Test Characterization** – Generic properties for Cell Tests. Basic information to represent the test, such as *Test Name, Test bench* or *Simulation Version* and the *Use Case.* This model is connected to the *Cell Test Procedures, Cell DoE Test Procedures , Cell Test Inputs* and *Cell Test Outputs* models, as each *Test* will need data structures to store this related data.
- **Cell DoE Test Procedures** – Configuration for each Test Procedure as output from the *DoE* for one test. Considers properties such as *Temperature, Voltage* or *Current*.
- **Cell Test Procedures** – Configuration for each standard Test Procedure suggested by the Digital Twin to the *DoE* for one test. Considers properties such as *Temperature, Voltage* or *Current*.

- **Cell Test Inputs** – Represents the inputs for each Cell Test. Considers *Time, Voltage* and *Current*.
- **Cell Test Outputs** – Represents the outputs for each Cell Test. Considers *Time, Temperature* and *Capacity*.

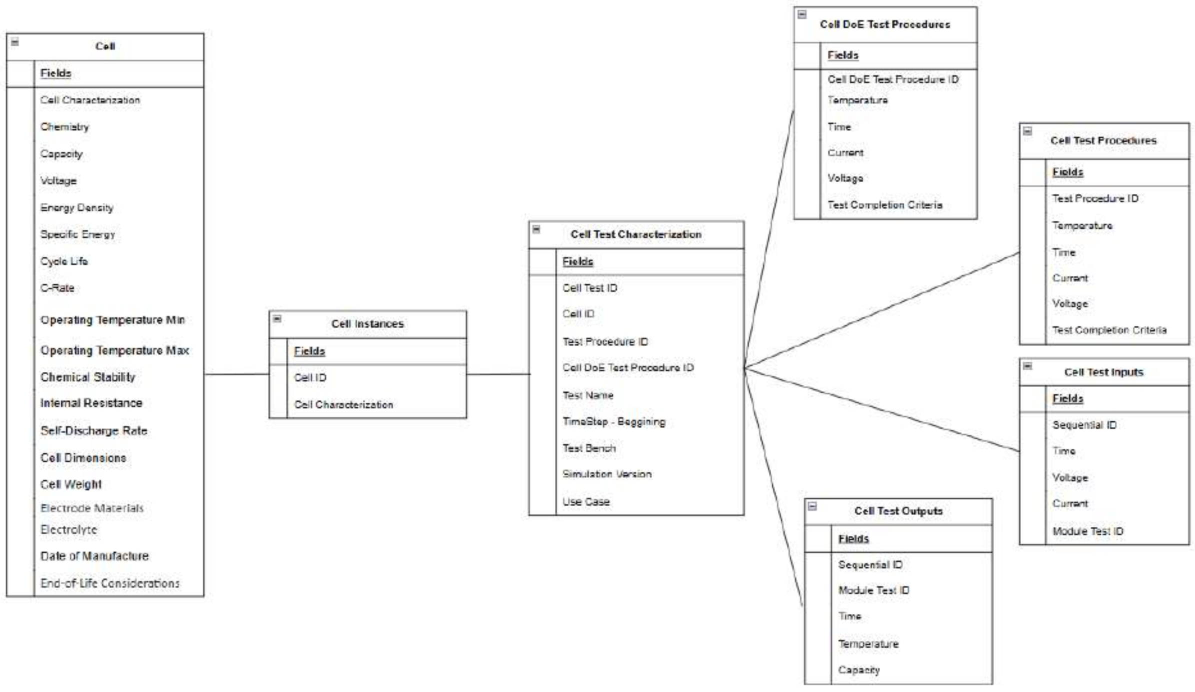


Figure 4 - Data Models schema for Cell Testing

All developed data schemas consider the *Use Case* for *Test Characterization,* this means that the structure is designed to absorb data relating to any Use Case, while still being agnostic enough to be able to consider extra Use Cases in the future, thus guaranteeing flexibility to adapt to market needs.

The code version of FASTEST Data Models can be found on Appendix A, and will be shared on Zenodo platform.

7.2. Digital Twin Models

Digital Twin Models are a specialized type of models designed for IoT applications that provides a standardized way to describe digital representations of IoT-enabled ecosystems, creating scalable and data-driven solutions. Digital Twin Models and Data Models can be used to complement each other in the same solution, as they fulfill different roles. Data Models provide a structured way to store and organize data, and Digital Twin Models bridge the gap for IoT and continuous data ingestion from streaming data sources, thus digitally reflecting a replica of the physical asset. Digital Twin Models, together with Data Models, form the backbone of the Digital Twin, guaranteeing a structure that can support different applications and frontends.

FASTEST Digital Twin system is based on Azure Platform, therefore Digital Twin Models are built under Azure Digital Twin Definition Language (DTDL), a JSON-LD-based schema language, and can be seen in Figure 5 as a Model Graph Visualization.

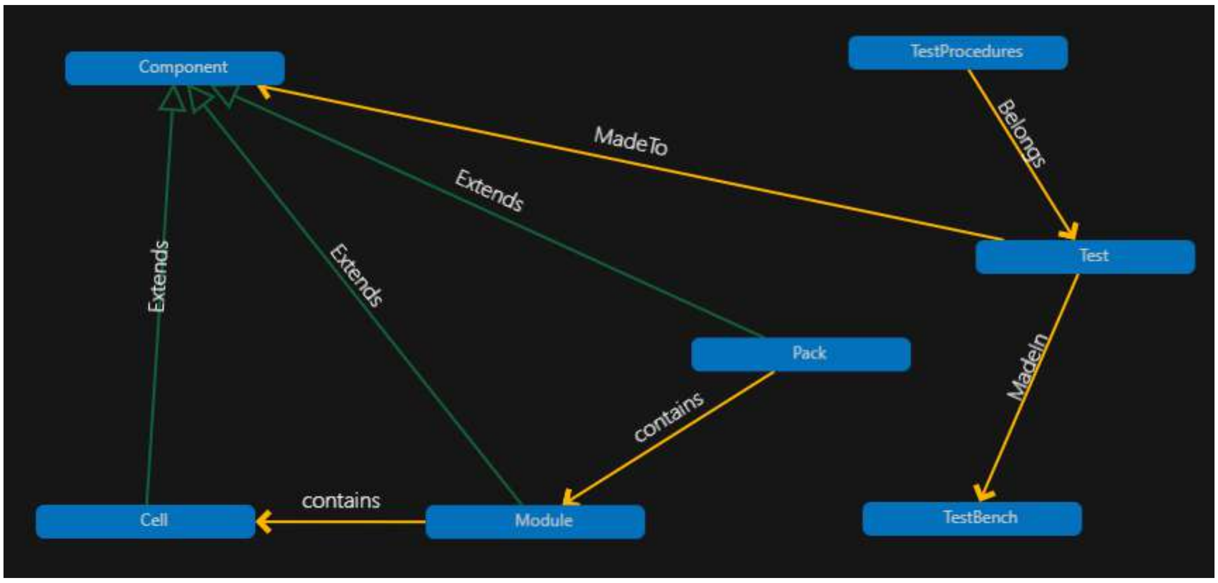
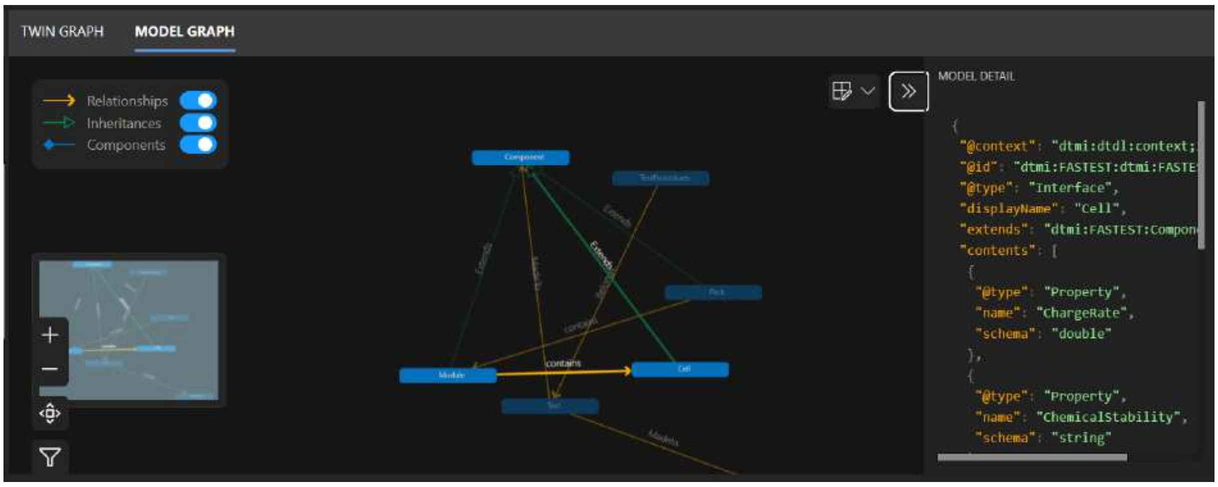


Figure 5 - Digital Twin DTDL Models represented in Azure Model Graph

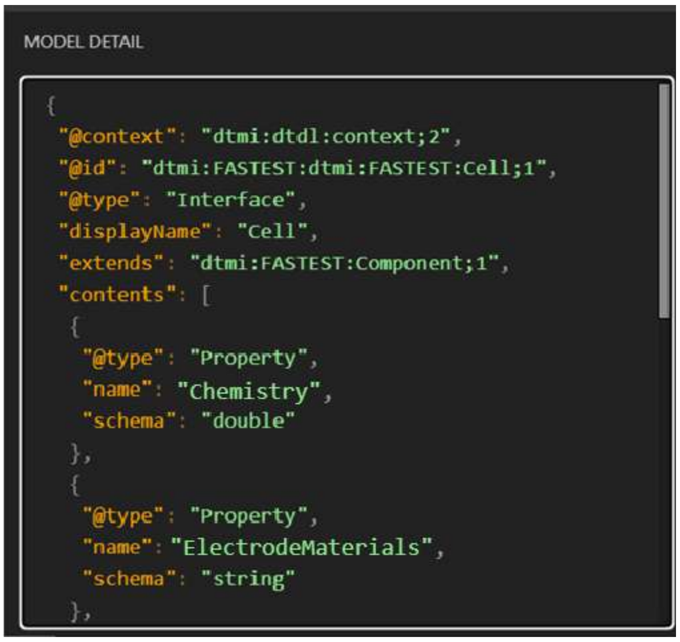
FASTEST Digital Twin Models consist of seven main classes and specific relationships to allow correct continuous data ingestion from cells, modules and battery pack tests. The relationships can be explained as the following, *Component* and *Cell/Module/Battery Pack* means that a Component is composed of one of these sub-elements. *Test* and *Component* have a relationship that indicates a Test is specifically designed or performed for a given Component. This is useful in scenarios where different components undergo distinct testing processes. *Test* and *TestBench* represent the environment where a Test is conducted. It shows that a Test occurs within a designated TestBench, emphasizing the importance of controlled testing conditions. It also includes the possibility of having physical and virtual tests within *TestBench* class. *Test* and *TestProcedures* are related as each Test will be executed under a set of specific Test Procedures provided by the DoE. These relationships help model/real-world synchronization within a Digital Twins environment, ensuring that digital representations accurately reflect the structure and behavior of physical and virtual entities.

Figure 6 and Figure 7 illustrate Azure view of Model and Relationship details. In this example it can be seen that Cell has a relationship of type *extend* with Component, and in the *contents* array it is seen the Cell Properties that are also detailed previously in Cell Characterization Data Model.



```
{
 "@context": "dtmi:dtdl:context;
 "@id": "dtmi:FASTEST:dtmi:FASTE
 "@type": "Interface",
 "displayName": "Cell",
 "extends": "dtmi:FASTEST:Compon
 "contents": [
  {
   "@type": "Property",
   "name": "ChargeRate",
   "schema": "double"
  },
  {
   "@type": "Property",
   "name": "ChemicalStability",
   "schema": "string"
```

Figure 6 - Digital Twin Model properties and details


MODEL DETAIL

```
{
 "@context": "dtmi:dtdl:context;2",
 "@id": "dtmi:FASTEST:dtmi:FASTEST:Cell;1",
 "@type": "Interface",
 "displayName": "Cell",
 "extends": "dtmi:FASTEST:Component;1",
 "contents": [
  {
   "@type": "Property",
   "name": "Chemistry",
   "schema": "double"
  },
  {
   "@type": "Property",
   "name":"ElectrodeMaterials",
   "schema": "string"
  },
```

Figure 7 - Cell Model and Relationship details

A rigorous specification of all the properties and relationships of digital assets leads to the construction of the Digital Twin backbone, preparing it for real-time data ingestion.

The following table provides a detailed description of the FASTEST Digital Twin Models, which full code version can be seen in Appendix A.

Type	Class	Name	Data Type
Property	Component	ID	String
Property	Component	Type	String
Property	Component	Capacity	Double
Property	Component	Voltage	Double
Property	Component	EnergyDensity	Double
Property	Component	SpecificEnergy	Double
Property	Component	CycleLife	String
Property	Component	InternalResistance	Double
Property	Component	OperatingTemperatureMin	Double
Property	Component	OperatingTemperatureMax	Double
Property	Component	DateOfManufacture	DateTime
Property	Component	Dimensions	Double
Property	Component	Weight	Double
Property	Cell	ChargeRate	Double
Property	Cell	ChemicalStability	String
Property	Cell	Chemistry	String
Property	Cell	ElectrodeMaterials	String
Property	Cell	Electrolyte	String
Property	Cell	SelfDischargeRate	Double
Property	Module	ListOfCellsIDs	String
Property	Module	CellArrangement	String
Property	Module	CellBalancing	String
Property	Module	ModuleConfiguration	String
Property	Module	NumberOfCells	Integer
Property	Pack	ListOfModulesIDs	String
Property	Pack	ModuleArrangement	String
Property	Pack	ThermalManagement	String
Property	Pack	NumberOfModules	Integer
Property	TestProcedures	TestingProceduresID	String
Property	TestProcedures	ExpectedResults	String
Property	TestProcedures	TestCompletionCriteria	String
Property	TestProcedures	TestExecutionSteps	String
Property	TestProcedures	TestPreRequisites	String
Property	TestBench	TestBenchID	String
Property	TestBench	TestBenchType	String
Property	Test	ComponentID	String
Property	Test	TestID	String

Property	Test	TestType	String
Property	Test	TestingProceduresID	String
Property	Test	TimeStepBeggining	String
Property	Test	SimulationVersion	String
Property	Test	TestBentchID	String
Telemetry	Test	Time	Time
Telemetry	Test	Voltage	Double
Telemetry	Test	Current	Double
Telemetry	Test	Temperature	String

Table 1 - Digital Twin Models

FASTEST Digital Twin Models are built for Azure Platform; however, their structure is technology-agnostic as it can without much effort be migrated to other Digital Twin Language to be integrated with different software platforms, or even be adapted to slightly different domains as battery testing. The scalable nature of Azure ensures that as battery technologies evolves, the Digital Twin framework can expand accordingly, making it a cost-effective and future-proof solution for large-scale battery testing systems.

7.3. Communication Model

One of the fundamental aspects of FASTEST is the communication between the internal Digital Twin modules and the external WP6 platform. Having aneffective communication will enable a correct integration between all the parties involved in FASTEST and establish structured messaging formats and protocols to ensure accurate information transfer. This requirement highlighted the need to develop a communication model, which governs how data is structured, transmitted, and interpreted, based on a technology-agnostic structure. This ensures that each party would be free to select the technology, and also so that in the future, after the project, the Digital Twin would not be tied to a specific technology, but would have a consolidated communication structure. This model utilizes lightweight messaging protocols that enable efficient real-time data exchange. The structure of these messages ensures that all relevant parameters, such as test identifiers, variables, and timestamped measurements, are consistently transmitted and processed.

The communication model designed for FASTEST considers hierarchical key and value sets that represent all the data required for effective communication between systems.Table 2 and Table 3 contains all the key-value pairs identified, outlining an example to better illustrate, in some cases with expected value sets, and in other cases with substructures within the initial key, as are related data.

Key	**Value [Example]**	**Expected Value**
Test_name	Overcharge	-

Test_type	Cell-level	-
Test_UUID	c61da39a-ace5-4f92-ad76-9a64516dc84d	-
Test_UUT	Cell-01	-
Test_bench	Physical	-
Test_procedures	T 30 60 90, 1C 5C 10C	-
Test_date	2024-08-05 10:00:00 UTC	-
Status	Start (Last update: 123)	-
Variables	See variables structure in Table 3	-

Table 2- Communication Model Keys, Values[Examples] and Expected Values when applied.

Key	Type	Unit	Timestamp [Example]	Value [Example]
Voltage	Output	mV	2024-08-05 12:00:00 UTC	1.0
Current	Input	mA	2024-08-05 12:05:00 UTC	2.0
Temperature	Output	°C	2024-08-05 12:05:00 UTC	25.0

Table 3 - Variables structure from Communication Model

The core elements of the FASTEST Communication Model include a test descriptor containing the test name, test type referring to cell, module or pack level, a universally unique identifier (UUID) to ensure test traceability, test UUT to specify the unit under test, the test bench as physical or virtual, a subset for the test procedures, a timestamp to record the test date, the status of the test, and a sub-structure for the variables, including the voltage, current and temperature, each one indicating if is of type Input or Output from the test, with the corresponding timestamp, unit and value. This nested structure ensures modularity and allows dynamic expansion for additional parameters. The entire message structure has been optimized to fulfill communication requirements for the Digital Twin, Laboratory Information Management Systems (LIMS), Physical Test Benches, Co-Simulation Platform and Design of Experiments (DoE) framework.

At this stage of the FASTEST project, basic communication tests have already been carried out that validate this Communication Model between WP5 and WP6 Platforms, thus giving a positive outlook on the interaction between them for the remaining activities of the project.

The full code version of the Communication Model can be seen in Appendix C.

7.4. Metadata Model

Metadata plays a critical role in Digital Twin systems, providing additional descriptive information, such as versioning, serial numbers, manufacturer details, creation timestamps, status tracking, etc., helping to manage and track digital twins efficiently in large-scale IoT systems. The Metadata Model is essential for managing and maintaining Digital Twin Models efficiently. It enables version control by tracking different iterations of the model, ensuring consistency and controlled updates. It also provides traceability and auditing by storing details like creation date, last update, and author, making it easier to track changes, debug issues, and ensure compliance. The model helps manage relationships by linking metadata with specific components, creating structured connections. Additionally, it supports data provenance and source tracking, ensuring the reliability and authenticity of the information used in the Digital Twin. By including descriptions and structured documentation, it enhances understanding for developers and users, making the model easier to interpret and use. Thus, the Metadata Model facilitates interoperability and integration with other systems, improving overall system efficiency and usability.

FASTEST Metadata Model defines standardized attributes that provide essential information about all Digital Twin assets. Table 4 contains all the key-value pairs identified as relevant metadata properties.

Property	Data Type
ModelID	String
ModelName	String
Version	String
CreationDate	DateTime
LastUpdated	DateTime
Author	String
Description	String
RelatedComponentID	String
DataSource	String

Table 4 - FASTEST Metadata Model

A well-defined metadata structure ensures consistency across Digital Twin and WP6 FASTEST platforms by categorizing information.

FASTEST Metadata Model considers properties such as ModelID a unique identifier assigned to the metadata model, ensuring it can be distinctly referenced in the system, ModelName, Version with a number of the model allowing for version control and tracking changes over time, CreationDate referencing a timestamp indicating when the metadata model was initially created helping with historical tracking, LastUpdated as the timestamp of the most recent update, Author as the name or identifier of the responsible for creating or modifying the metadata model ensuring traceability, Description as a brief textual summary explaining the purpose, function or scope of the metadata model, RelatedComponentID as an identifier linking the metadata model to a specific component or entity within the Digital Twin system, establishing relationships between models, and DataSource as the origin or source of data used in the metadata model, providing transparency on where the information is derived from.

By integrating these metadata properties, Digital Twin frameworks enhance data integrity and facilitate future reuse and collaboration across other applications.

The full code version of the Metadata Model can be seen in Appendix D.

8. Conclusion

Deliverable 5.2 reports the output of Task 5.2 in the form of Data Models, Digital Twin Models, Communication and Metadata Models. These are the foundation of the FASTEST Digital Twin and have been conceptualised in models with a sufficient level of abstraction to be technology-agnostic and, to a certain extent, application-agnostic, requiring only a minor degree of effort to adapt them to a testing reality in another domain. These Models together with the entire technical description from the perspective of the application system developed, will allow to take another step towards Objective 3 of the FASTEST project, which aims to instantiate 3 Digital Twins, one dedicated to each Use Case, automotive, stationary and off-road.

About the work carried out within T5.2 scope, and as it is reported on Chapter 4, the Ontology resulted from T5.1 was used as input for this task. It leveraged the construction of a guiding thread through the basic characterisation of the physical assets being tested, as well as a prior specification of relevant points relating to the tests. The evolution of the Ontology into structured Models, provided a framework for categorizing and defining core elements. This, together with inputs from WP2, WP3, WP4 and WP6 partners made it possible to take the first steps towards the focus of this task, the development of the Models. This task was based on a series of activities, resulting in a set of Models with different purposes that complement each other and form the conceptual backbone of FASTEST Digital Twin. These Models support the architecture and specific software modules that are detailed in D5.3 Integration plan for Digital Twin on the platform.

Data Models serve as the backbone for structuring and organizing the data needed for Digital Twin operations, while Digital Twin Models define the virtual representations that interact with other platforms. The Communication Model ensures efficient data exchange, enabling data synchronization and decision-making. The Metadata Model further enhances model traceability, governance, and usability by providing essential information about version control, relationships, and data provenance.

By establishing these foundational and agnostic elements, this report provides a roadmap for implementing and refining Digital Twin solutions, specially applied to the battery testing domain. The evolving nature of Digital Twins need continuous development and adaptation to meet the growing demands of various industries. These models may need some refining or adjustment until the end of FASTEST to consolidate their correspondence with the needs of the project.

9. References

Duan, H., Gao, S., Yang, X., & Li, Y. (2022). The development of a digital twin concept system. *Digital Twin*, 10.

Glaessgen, E. H., & Stargel, D. S. (sd). The Digital Twin Paradigm for Future NASA and U.S. Air Force Vehicles. *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference&amp;lt;BR&amp;gt;20th AIAA/ASME/AHS Adaptive Structures Conference&amp;lt;BR&amp;gt;14th AIAA.*

Grieves, M., & Vickers, J. (2016). Digital Twin: Mitigating unpredictable, undesirable emergent behavior in complex systems. *Transdisciplinary Perspectives on Complex Systems*, 85-113.

Gupta, A., Khare, A., & Rawat, R. (2023). Digital Twin and digital twin–driven manufacturing. *Digital Twin for Smart Manufacturing*, 1-19.

Rosen, R., von Wichert, G., Lo, G., & Bettenhausen, K. D. (2015). About the importance of autonomy and digital twins for the future of manufacturing. *IFAC-PapersOnLine*, 567-572.

Rosen, R., von Wichert, G., Lo, G., & Bettenhausen, K. D. (2015). About the importance of autonomy and digital twins for the future of manufacturing. *IFAC-PapersOnLine*, 567-572.

Tao, F., Cheng, J., Qi, Q., Zhang, M., Zhang, H., & Sui, F. (2017). Digital twin-driven product design, manufacturing and service with Big Data. *The International Journal of Advanced Manufacturing Technology*, 3563-3576.

Appendix A: Data Models

This appendix provides the Data Models, which specify the data entities, fields and relationships required for modeling battery testing and validation.

```
CREATE TABLE Pack (
    PackID INT PRIMARY KEY,
    PackCharacterization TEXT,
    PackConfiguration TEXT,
    PackCapacity FLOAT,
    PackVoltage FLOAT,
    PackEnergyDensity FLOAT,
    PackSpecificEnergy FLOAT,
    NumberOfModules INT,
    ModuleArrangement TEXT,
    ModuleBalancing TEXT,
    ThermalManagement TEXT,
    CycleLife INT,
    OperatingTemperatureMin FLOAT,
    OperatingTemperatureMax FLOAT,
    InternalResistance FLOAT,
    PackWeight FLOAT,
    PackDimensions TEXT,
    DateOfManufacture DATE,
    EndOfLifeConsiderations TEXT
);

CREATE TABLE Module (
    ModuleID INT PRIMARY KEY,
    ModuleCharacterization TEXT,
    ModuleConfiguration TEXT,
    ModuleCapacity FLOAT,
    ModuleVoltage FLOAT,
    ModuleEnergyDensity FLOAT,
    ModuleSpecificEnergy FLOAT,
    NumberOfCells INT,
    CellArrangement TEXT,
    CellBalancing TEXT,
    ThermalManagement TEXT,
    CycleLife INT,
    OperatingTemperatureMin FLOAT,
    OperatingTemperatureMax FLOAT,
    InternalResistance FLOAT,
    ModuleWeight FLOAT,
    ModuleDimensions TEXT,
    DateOfManufacture DATE,
    EndOfLifeConsiderations TEXT
);
```

```sql
CREATE TABLE Cell (
    CellID INT PRIMARY KEY,
    CellCharacterization TEXT,
    Chemistry TEXT,
    Capacity FLOAT,
    Voltage FLOAT,
    EnergyDensity FLOAT,
    SpecificEnergy FLOAT,
    CycleLife INT,
    CRate FLOAT,
    OperatingTemperatureMin FLOAT,
    OperatingTemperatureMax FLOAT,
    ChemicalStability TEXT,
    InternalResistance FLOAT,
    SelfDischargeRate FLOAT,
    CellWeight FLOAT,
    ElectrodeMaterials TEXT,
    Electrolyte TEXT,
    DateOfManufacture DATE,
    EndOfLifeConsiderations TEXT
);

CREATE TABLE PackInstances (
    PackInstanceID INT PRIMARY KEY,
    PackID INT,
    PackCharacterization TEXT,
    FOREIGN KEY (PackID) REFERENCES Pack(PackID)
);

CREATE TABLE ModuleInstances (
    ModuleInstanceID INT PRIMARY KEY,
    ModuleID INT,
    ModuleCharacterization TEXT,
    FOREIGN KEY (ModuleID) REFERENCES Module(ModuleID)
);

CREATE TABLE CellInstances (
    CellInstanceID INT PRIMARY KEY,
    CellID INT,
    CellCharacterization TEXT,
    FOREIGN KEY (CellID) REFERENCES Cell(CellID)
);

CREATE TABLE PackTestCharacterization (
    PackTestID INT PRIMARY KEY,
    PackID INT,
    TestProcedureID INT,
    PackDoETestProcedureID INT,
    TestName TEXT,
    TimestampBeginning DATETIME,
    TestBench TEXT,
    SimulationVersion TEXT,
    UseCase TEXT,
    FOREIGN KEY (PackID) REFERENCES Pack(PackID)
);

CREATE TABLE PackTestInputs (
    SequentialID INT PRIMARY KEY,
    PackTestID INT,
    Time FLOAT,
    Voltage FLOAT,
    Current FLOAT,
    FOREIGN KEY (PackTestID) REFERENCES PackTestCharacterization(PackTestID)
);

CREATE TABLE PackTestOutputs (
    SequentialID INT PRIMARY KEY,
    PackTestID INT,
    Time FLOAT,
    Temperature FLOAT,
    Capacity FLOAT,
    FOREIGN KEY (PackTestID) REFERENCES PackTestCharacterization(PackTestID)
);

CREATE TABLE PackTestProcedures (
    TestProcedureID INT PRIMARY KEY,
    Temperature FLOAT,
    Time FLOAT,
    Current FLOAT,
    Voltage FLOAT,
    TestCompletionCriteria TEXT
);

CREATE TABLE PackDoETestProcedures (
    PackDoETestProcedureID INT PRIMARY KEY,
    Temperature FLOAT,
    Time FLOAT,
    Current FLOAT,
    Voltage FLOAT,
    TestCompletionCriteria TEXT
);

CREATE TABLE ModuleTestCharacterization (
    ModuleTestID INT PRIMARY KEY,
    ModuleID INT,
    TestProcedureID INT,
    ModuleDoETestProcedureID INT,
    TestName TEXT,
    TimestampBeginning DATETIME,
    TestBench TEXT,
    SimulationVersion TEXT,
    UseCase TEXT,
    FOREIGN KEY (ModuleID) REFERENCES Module(ModuleID)
);

CREATE TABLE ModuleTestInputs (
    SequentialID INT PRIMARY KEY,
    ModuleTestID INT,
    Time FLOAT,
    Voltage FLOAT,
    Current FLOAT,
    FOREIGN KEY (ModuleTestID) REFERENCES
ModuleTestCharacterization(ModuleTestID)
);

CREATE TABLE ModuleTestOutputs (
    SequentialID INT PRIMARY KEY,
    ModuleTestID INT,
    Time FLOAT,
    Temperature FLOAT,
    Capacity FLOAT,
    FOREIGN KEY (ModuleTestID) REFERENCES
ModuleTestCharacterization(ModuleTestID)
);

CREATE TABLE ModuleTestProcedures (
    TestProcedureID INT PRIMARY KEY,
    Temperature FLOAT,
    Time FLOAT,
    Current FLOAT,
    Voltage FLOAT,
    TestCompletionCriteria TEXT
);

CREATE TABLE ModuleDoETestProcedures (
    ModuleDoETestProcedureID INT PRIMARY KEY,
    Temperature FLOAT,
    Time FLOAT,
    Current FLOAT,
    Voltage FLOAT,
    TestCompletionCriteria TEXT
);

CREATE TABLE CellTestCharacterization (
    CellTestID INT PRIMARY KEY,
    CellID INT,
    TestProcedureID INT,
    CellDoETestProcedureID INT,
    TestName TEXT,
    TimestampBeginning DATETIME,
    TestBench TEXT,
    SimulationVersion TEXT,
    UseCase TEXT,
    FOREIGN KEY (CellID) REFERENCES Cell(CellID)
);

CREATE TABLE CellTestInputs (
    SequentialID INT PRIMARY KEY,
    CellTestID INT,
    Time FLOAT,
    Voltage FLOAT,
    Current FLOAT,
    FOREIGN KEY (CellTestID) REFERENCES CellTestCharacterization(CellTestID)
);

CREATE TABLE CellTestOutputs (
    SequentialID INT PRIMARY KEY,
    ModuleTestID INT,
    Time FLOAT,
    Temperature FLOAT,
    Capacity FLOAT,
    FOREIGN KEY (ModuleTestID) REFERENCES
ModuleTestCharacterization(ModuleTestID)
);

CREATE TABLE CellTestProcedures (
    CellDoETestProcedureID INT PRIMARY KEY,
    Temperature FLOAT,
    Time FLOAT,
    Current FLOAT,
    Voltage FLOAT,
    TestCompletionCriteria TEXT
);
```

Appendix B: Digital Twin Models

This appendix provides the DTDL (Digital Twins Definition Language) model definitions. These specify the core entities, properties, relationships, and telemetry required for modeling battery testing and validation.

1. Component

This class defines an individual component atribute and telemetry data:

```
{
  "@context": "dtmi:dtdl:context;2",
  "@id": "dtmi:FASTEST:Component;1",
  "@type": "Interface",
  "displayName": "Component",
  "contents": [
    {
      "@type": "Property",
      "name": "ID",
      "schema": "string"
    },
    {
      "@type": "Property",
      "name": "Type",
      "schema": "string"
    },
    {
      "@type": "Property",
      "name": "Capacity",
      "schema": "double"
    },
    {
      "@type": "Property",
      "name": "Voltage",
      "schema": "double"
    },
    {
      "@type": "Property",
      "name": "EnergyDensity",
      "schema": "double"
    },
    {
      "@type": "Property",
      "name": "SpecificEnergy",
      "schema": "double"
    },
    {
      "@type": "Property",
      "name": "CycleLife",
      "schema": "string"
    },
    {
      "@type": "Property",
      "name": "InternalResistance",
      "schema": "double"
    },
    {
      "@type": "Property",
      "name": "OperatingTemperatureMin",
      "schema": "double"
    },
    {
      "@type": "Property",
      "name": "OperatingTemperatureMax",
      "schema": "double"
    },
    {
      "@type": "Property",
      "name": "DateOfManufacture",
      "schema": "dateTime"
    },
    {
      "@type": "Property",
      "name": "Dimensions",
      "schema": "double"
    },
    {
      "@type": "Property",
      "name": "Weight",
      "schema": "double"
    },
    {
      "@type": "Property",
      "name": "EndOfLifeConsiderations",
      "schema": "string"
    }
  ]
}
```

1.1 Cell

This subclass of Component defines an individual battery cell's attributes and telemetry data:

```
{
  "@context": "dtmi:dtdl:context;2",
  "@id": "dtmi:FASTEST:dtmi:FASTEST:Cell;1",
  "@type": "Interface",
  "displayName": "Cell",
  "extends": "dtmi:FASTEST:Component;1",
  "contents": [
    {
      "@type": "Property",
      "name": "ChargeRate",
      "schema": "double"
    },
    {
      "@type": "Property",
      "name": "ChemicalStability",
      "schema": "string"
    },
    {
      "@type": "Property",
      "name": "Chemistry",
      "schema": "string"
    },
    {
      "@type": "Property",
      "name": "ElectrodeMaterials",
      "schema": "string"
    },
    {
      "@type": "Property",
      "name": "Electrolyte",
      "schema": "string"
    },
    {
      "@type": "Property",
      "name": "SelfDischargeRate",
      "schema": "double"
    }
  ]
}
```

1.2 Module

This subclass of Component defines an individual battery module attributes and telemetry data:

```
{
  "@context": "dtmi:dtdl:context;2",
  "@id": "dtmi:FASTEST:dtmi:FASTEST:Module;1",
  "@type": "Interface",
  "displayName": "Module",
  "extends": "dtmi:FASTEST:Component;1",
  "contents": [
    {
      "@type": "Property",
      "name": "ListOfCellsIDs",
      "schema": "string"
    },
    {
      "@type": "Property",
      "name": "CellArrangement",
      "schema": "string"
    },
    {
      "@type": "Property",
      "name": "CellBalancing",
      "schema": "string"
    },
    {
      "@type": "Property",
      "name": "ModuleConfiguration",
      "schema": "string"
    },
    {
      "@type": "Property",
      "name": "NumberOfCells",
      "schema": "integer"
    },
    {
      "@type": "Relationship",
      "name": "contains",
      "displayName": "contains",
      "target": "dtmi:FASTEST:dtmi:FASTEST:Cell;1"
    }
  ]
}
```

1.3 Pack

This subclass of Component defines an individual battery pack attributes and telemetry data:

```
{
    "@context": "dtmi:dtdl:context;2",
    "@id": "dtmi:FASTEST:Pack;1",
    "@type": "Interface",
    "displayName": "Pack",
    "extends": "dtmi:FASTEST:Component;1",
    "contents": [
        {
            "@type": "Property",
            "name": "ListOfModulesIDs",
            "schema": "string"
        },
        {
            "@type": "Property",
            "name": "ModuleArrangement",
            "schema": "string"
        },
        {
            "@type": "Property",
            "name": "ThermalManagement",
            "schema": "string"
        },
        {
            "@type": "Property",
            "name": "NumberOfModules",
            "schema": "integer"
        },
        {
            "@type": "Relationship",
            "name": "contains",
            "displayName": "contains",
            "target": "dtmi:FASTEST:dtmi:FASTEST:Module;1"
        }
    ]
}
```

2. Test

This class defines an individual test attributes and telemetry data:

```
{
    "@context": "dtmi:dtdl:context;2",
    "@id": "dtmi:FASTEST:dtmi:FASTEST:Test;1",
    "@type": "Interface",
    "displayName": "Test",
    "contents":[
        {
            "@type": "Property",
            "name": "ComponentID",
            "schema": "string"
        },
        {
            "@type": "Property",
            "name": "TestID",
            "schema": "string"
        },
        {
            "@type": "Property",
            "name": "TestType",
            "schema": "string"
        },
        {
            "@type": "Property",
            "name": "TestingProceduresID",
            "schema": "string"
        },
        {
            "@type": "Property",
            "name": "TimeStepBeggining",
            "schema": "date"
        },
        {
            "@type": "Property",
            "name": "SimulationVersion",
            "schema": "string"
        },
        {
            "@type": "Property",
            "name": "TestBentch",
            "schema": "string"
        },
        {
            "@type": "Telemetry",
            "name": "Time",
            "schema": "time"
        },
        {
            "@type": "Telemetry",
            "name": "Voltage",
            "schema": "double"
        },
        {
            "@type": "Telemetry",
            "name": "Current",
            "schema": "double"
        },
        {
            "@type": "Telemetry",
            "name": "Temperature",
            "schema": "string"
        },
      {
        "@type": "Relationship",
        "name": "MadeTo",
        "displayName": "MadeTo",
        "target": "dtmi:FASTEST:Component;1"
      },
        {
            "@type": "Relationship",
            "name": "MadeIn",
            "displayName": "MadeIn",
            "target": "dtmi:FASTEST:dtmi:FASTEST:TestBench;1"
        }
    ]
}
```

4. Test Bench

This class defines an individual test bench attributes and telemetry data:

```
{
    "@context": "dtmi:dtdl:context;2",
    "@id": "dtmi:FASTEST:dtmi:FASTEST:TestBench;1",
    "@type": "Interface",
    "displayName": "TestBench",
    "contents":[
        {
            "@type": "Property",
            "name": "TestBenchID",
            "schema": "string"
        },
        {
            "@type": "Property",
            "name": "TestBenchType",
            "schema": "string"
        }
    ]
}
```

5. Test Procedures

This class defines an individual testing procedure attributes and telemetry data:

```
{
    "@context": "dtmi:dtdl:context;2",
    "@id": "dtmi:FASTEST:dtmi:FASTEST:TestProcedures;1",
    "@type": "Interface",
    "displayName": "TestProcedures",
  "contents": [
    {
      "@type": "Property",
      "name": "TestingProceduresID",
      "schema": "string"
    },
    {
      "@type": "Property",
      "name": "ExpectedResults",
      "schema": "string"
    },
    {
      "@type": "Property",
      "name": "TestCompletionCriteria",
      "schema": "string"
    },
    {
      "@type": "Property",
      "name": "TestExecutionSteps",
      "schema": "string"
    },
    {
      "@type": "Property",
      "name": "TestPreRequisites",
      "schema": "string"
    },
    {
      "@type": "Relationship",
      "name": "Belongs",
      "displayName": "Belongs",
      "target": "dtmi:FASTEST:dtmi:FASTEST:Test;1"
    }
  ]
}
```

Appendix C: Communication Model

This appendix provides the Communication Model for FASTEST, specifying the core parameters to establish communication with DT and between internal DT modules.

```
Topic: /fastest_tests_testProcedures
Message:{
     "test_name": "Overcharge",
     "test_type":  "Cell-level",
     "test_UUID": "c61da39a-ace5-4f92-ad76-9a64516dc84d",
     "test_UUT":  "Cell-01",
     "test_bench": "Virtual/Physical",
     "variables": [{
            "name": "Voltage",
            "type": "Output",
            "unit": "mV",
            "values": [ {
                "timestamp": 12345678,
                "value": 1.0
                        },
                        {
                            "timestamp": 12345678,
                            "value": 2.0
                            },
                            ]
                         },
                         {
                            "name": "Current",
                            "type": "Input",
                            "unit": "mA",
                            "values": [ {
                                "timestamp": 12345678,
                                "value": 1.0
                                    },
                                    {
                                        "timestamp": 12345678,
                                        "value": 2.0
                                        }
                            ]
        }],
    "test_procedures":"T 30 60 90, 1C 5C 10C",
    "test_date": "2024-08-05 10:00:00 UTC",
    "status": {
        "value": "Start",
        "last_update": 123
    }}
```

Appendix D: Metadata Model

This appendix provides the Metadata Model for FASTEST,specifying the backbone properties that characterize each Digital Twin Model.

```
{
  "@context": "dtmi:dtdl:context;2",
  "@id": "dtmi:FASTEST:Metadata;1",
  "@type": "Interface",
  "displayName": "Metadata",
  "contents": [
    {
      "@type": "Property",
      "name": "ModelID",
      "schema": "string"
    },
    {
      "@type": "Property",
      "name": "ModelName",
      "schema": "string"
    },
    {
      "@type": "Property",
      "name": "Version",
      "schema": "string"
    },
    {
      "@type": "Property",
      "name": "CreationDate",
      "schema": "dateTime"
    },
    {
      "@type": "Property",
      "name": "LastUpdated",
      "schema": "dateTime"
    },
    {
      "@type": "Property",
      "name": "Author",
      "schema": "string"
    },
    {
      "@type": "Property",
      "name": "Description",
      "schema": "string"
    },
    {
      "@type": "Property",
      "name": "RelatedComponentID",
      "schema": "string"
    },
    {
      "@type": "Property",
      "name": "DataSource",
      "schema": "string"
    }
  ]
}
```