A framework to implement digital twin functionalities into web applications

WJ Kilian

orcid.org/0000-0002-9722-3627

Dissertation accepted in fulfilment of the requirements for the degree *Master of Engineering in Computer and Electronic Engineering* at the North-West University

Supervisor: Dr SGJ van Niekerk

Co-supervisor: Dr JC Vosloo

Graduation: May 2022

Student number: 37209469

Abstract

Title:	A framework to implement digital twin functionalities into web applications
Author:	Wilhelm Johannes Kilian
Supervisor:	Dr SGJ van Niekerk
Co-supervisor:	Dr JC Vosloo
Degree:	Master of Engineering in Computer and Electronic Engineering
Keywords:	Digital twin, web application, cross-platform, scalability, modular

Digital Twins provide instant physical context to data in complex engineering systems such as mines. They assist with mining operations, performance analysis, planning, and decision making when integrated with sensor data. However, these Digital Twins are computationally intensive and typically require powerful desktop computers to be used instead of much less powerful mobile devices.

Remote access to Digital Twin functionalities can be provided to many users through web applications. However, the constraints of web applications are less available computing power on devices (such as smartphones) as well as the high latency and cost of data transfer over the web. Hence, a need exists to optimise Digital Twin data models for web applications to enable cross-platform access to Digital Twin functionalities.

A framework was created which consists of an optimisation methodology for data models and a methodology to develop web applications. The optimisation methodology consists of analysing existing data models, a modular design methodology, and a compression method for data models. The methodology for the development of web applications forms the second half of the framework.

The results obtained were the reduction in the required Digital Twin data model size by more than 86% of the original size and a web application that was implemented to enable cross-platform access to Digital Twin functionalities. The optimisation methodology was used to store as much information in as limited data as possible while enabling modular development of the model. There were significant framerate improvements for the system on mobile devices when applying the methodology for the development of the web application. This resulted in an average of 60 frames per second for the three case studies used.

Due to the reduction in complexity and size of the data model as well as the performance of the web application, access to Digital Twin functionalities were enabled for many users with mobile devices such as smartphones. The framework created can be applied to Digital Twin data models in many different industries, such as mining and manufacturing industries, to enable remote access for users to Digital Twin functionalities.

Acknowledgements

To my love, Nerinda, your never-ending support and love enabled me to write this dissertation. You are an inseparable part of my life and I am immeasurably thankful to have met you. You will always be my first priority.

To my parents, Martie and Thomas, I am the man today because of how you raised me. Thank you for all the lessons you have taught me, easy and tough. You showed me how to build a family.

To Oupa Wilhelm, I miss you terribly. You showed me how years of hard work and commitment will build a home in which your family can prosper. You taught me the joy of working with my hands and to see my garden and life grow as a result of it. Thank you for the person you were in my life, you have no idea what you meant to me.

To Sybrand, my supervisor and mentor, thank you for all of your wisdom you imparted on me, not only professionally, but in my personal life as well. Thank you for the colleague and friend you have become.

To Jan Vosloo, my initial supervisor then co-supervisor, thank you for all of your contributions to this study. Your guidance at the start of this study formed a significant part of it.

To Prof. E. H. Mathews, thank you for the opportunity to do my Masters at CRCED Pretoria. Finally, thank you to ETA Operations (Pty) Ltd, for financial support to complete this study.

Contents

List of Figures

List of Tables

Table of abbreviations

AAS	Asset Administration Shell
API	Application Programming Interface
AR	Augmented Reality
BIM	Building Information Modelling
CAD	Computer-Aided Design
CPPS	Cyber-Physical Production System
CPS	Cyber-Physical System
C2PS	Cloud-based Cyber-Physical System
DT	Digital Twin
DXF	Drawing interchange format
FPS	Frames per second
GDTA	Generic Digital Twin Architecture
GIS	Geographic Information System
GML	Geography Markup Language
HMI	Human Machine Interface
IFC	Industry Foundation Classes
IoT	Internet of Things
IT	Integration test
JSON	JavaScript Object Notation
LOD	Level of detail
PDCA	Plan, Do, Check, Act
PDF	Portable Document Format
OS	Operating System
SCADA	Supervisory Control And Data Acquisition
VR	Virtual Reality
VREDI	Virtual Representation for Digital Twins
2D	Two-Dimensional
3D	Three-Dimensional

1 Background and existing studies

1.1 Introduction

1.1.1 Preamble

Complex engineering systems such as mines require means to visualise the systems that form part of it. Visualisation of complex systems gives physical context to those who design, analyse, and plan for changes to these systems. Physical context in any systems analysis and design procedure forms a crucial part of the process to identify problem areas and opportunities for improvement.

There are many different visualisation methods, such as physical drawings of a system's layout, Computer-Aided Design (CAD) drawings, and Digital Twins (DT) [1]. Each method offers unique advantages as well as shortcomings, however, DTs offer the unique advantage of providing instant physical context to data in complex engineering systems [2].

These DTs are, however, computationally intensive and typically require powerful desktop computers to be used instead of much less powerful mobile devices [3].

1.1.2 The Digital Twin

One definition of a DT is given by Wagner et al. [1] as consisting of three parts, namely the physical system, the digitised virtual system, and the data that tie the two together. Wagner et al. focused on the application of DTs in the manufacturing industry, however, the definition remains valid for most complex engineering systems [2–4].

DTs provide intuitive three-dimensional (3D) context of data within a complex system. Compared to, e.g. well-known Supervisory Control and Data Acquisition (SCADA) systems, the physical context is not as apparent as in a DT. These SCADA systems provide only a two-dimensional (2D) representation of the system, and as such, they do not provide full spatial context to the data.

Consider Figure 1.1 where a SCADA system interface example is shown. Most of the component's sensor data in this relatively simple system example can be shown to the user. The data is, however, without full physical context as each component is not in its true geometric place relative to all other components.

The components representing the system in Figure 1.1 have been scaled such that multiple graphs and system variables can be displayed on the interface. Basic relations between

components have been indicated by arrows and descriptions between components.

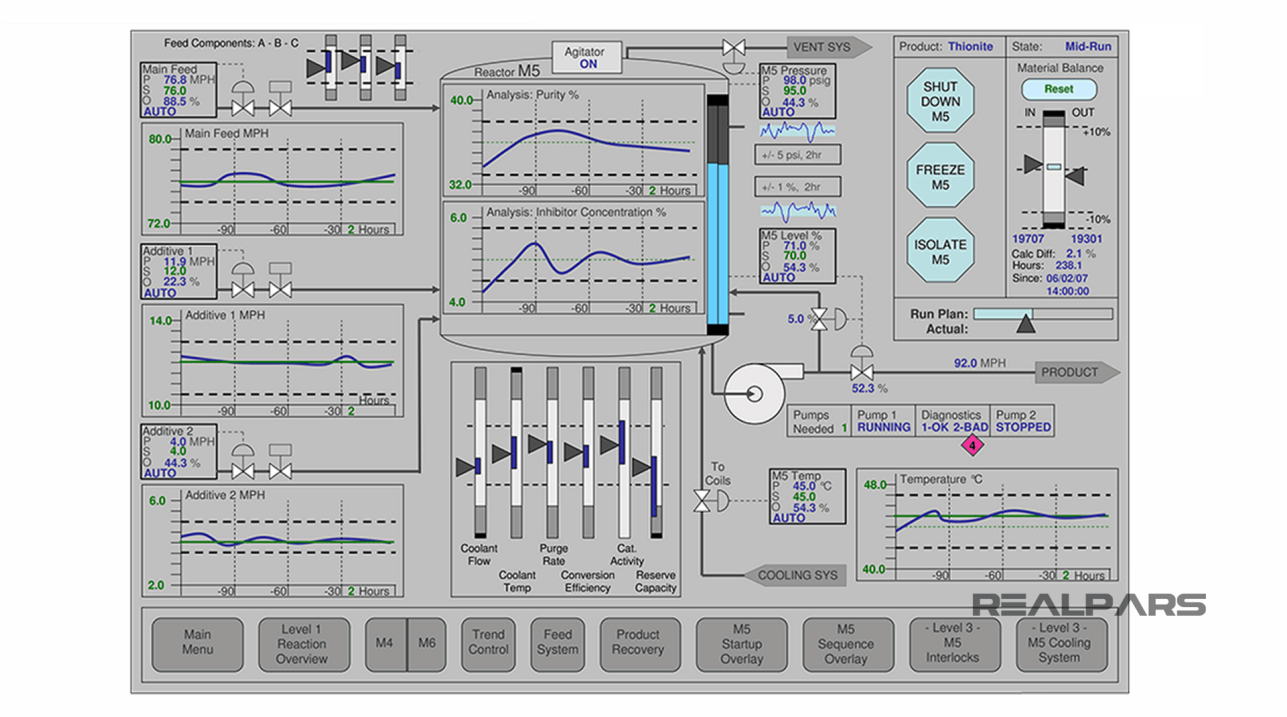


Figure 1.1: Example of a SCADA interface[1].

Although this interface may contain much information, the full physical context of each component within the system cannot be seen within the interface. Figure 1.2 shows an example of a DT interface for a pump. This example gives much more detailed physical context to each sensor's data when compared to the example of a SCADA system in Figure 1.1.

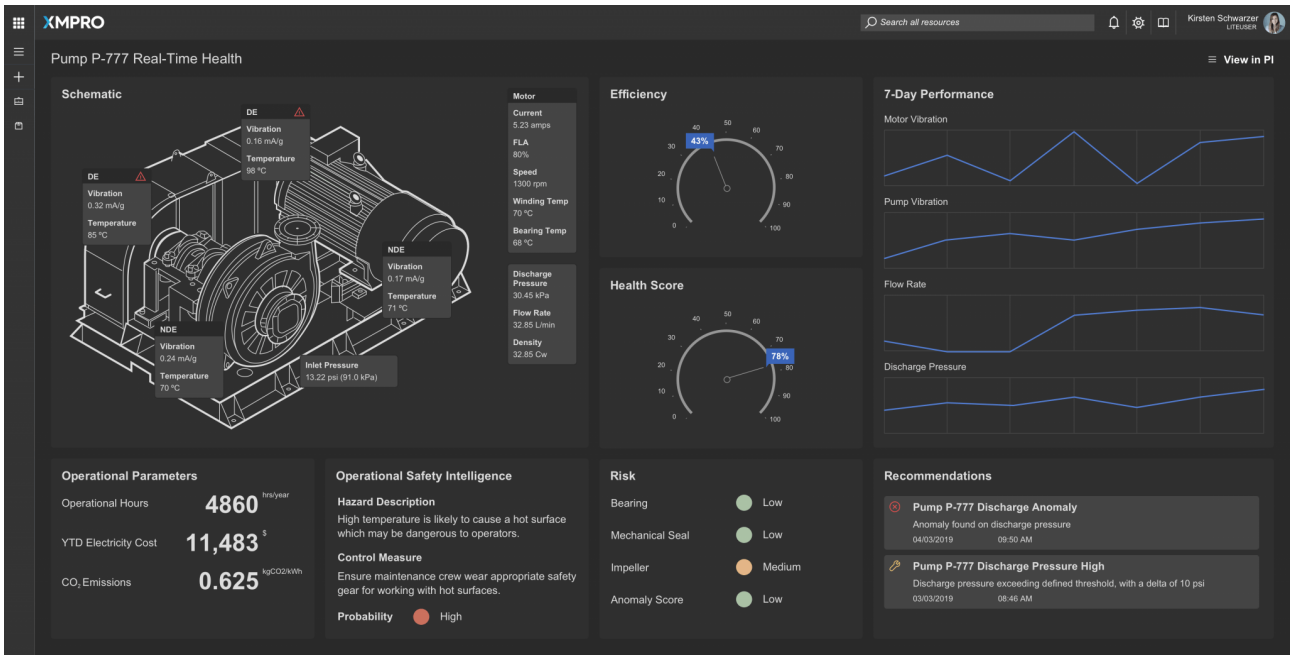


Figure 1.2: Example of a DT interface[2].

[1] Adapted from https://realpars.com/high-performance-hmi/.
[2] Taken from https://xmpro.com/digital-twins-the-ultimate-guide/

The data from the sensors within the system have been displayed in various ways on the DT interface, such as the pop-up graphs floating over the 3D model and the 7-day performance graphs in the rightmost portion of the interface.

The full physical context provided by DTs aid users in understanding the system more comprehensively for improvements to be made to system performance. Some of the advantages of DTs are:

- Detailed visual physical context of each component within the system [5],
- 3D interaction opportunities [6], and
- Live data integration [7].

Utilising these three advantages enable technical users to interpret the system performance and view the system itself, aiding them in further designs and improvements of the system. These three advantages also enable clients to view system performance and plan for future changes and improvements.

The simplified DT system shown in Figure 1.3 describes four of the core components of a DT. Firstly, the digital representation of the physical system that commonly consists of a digital model describing the physical properties of the system for which the DT is being created. The second component is the measurement data collected from the sensors within the system, e.g. water flow over time through a pipe.

The integration of the digital representation and the sensor data as well as various calculations and analysis using the collected data forms the third component. The final component is the 3D interactive user interface consisting of the 3D model of the system and the collected and analysed sensor data.

These four components have been placed into three main research categories in Figure 1.3, namely, the DT data model, the DT reference architecture, and lastly, the DT user interface.

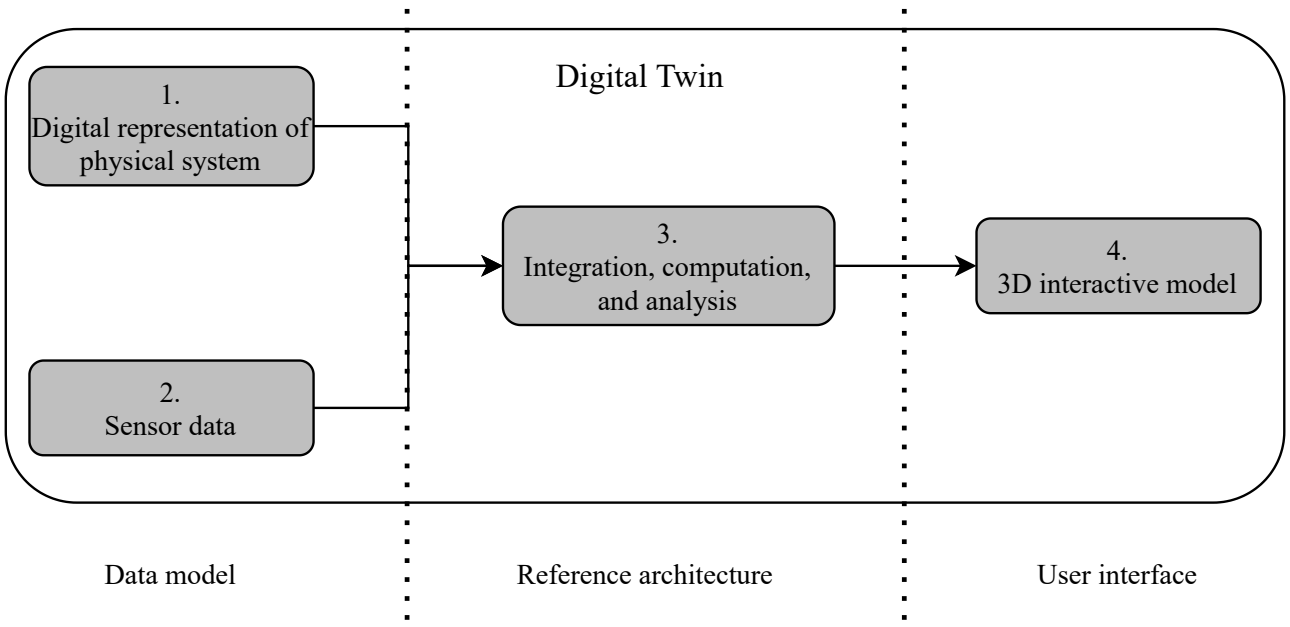


Figure 1.3: Core DT system.

1.1.3 Web applications

Web applications are one of the many different software applications that enable access to the user interface of a system to a client [8]. One of the most appealing advantages of using web applications is that any device that supports a web browser (as most modern devices do [9]) can access the functionalities provided. Another advantage is that no software installation by the user is required as web browsers are installed by default on almost all desktops, laptops, and smartphones.

Native applications are another type of software application which are developed for specific platforms [10], [11] (e.g. Android smartphones). It is also typically required that the user must install the software before they can access it (if it is not installed by default from the manufacturer). Therefore, web applications are more accessible to users than native applications in that they do not need to install software to access the functionalities.

The cross-platform property provides further advantages besides a system being accessible from most devices. It means that the development of only one application is required for the functionalities to be accessible on many different types of devices [8]. This significantly reduces development time in comparison with developing native applications, which require separate applications, i.e. development time, per type of device.

There are, however, a few disadvantages of using web applications when compared to native applications. One of which is that web applications are not as performant as native applications, as web applications are restricted from certain operating system (OS) functionalities which improve performance on the host device [11].

The compromise between shortened development time and performance losses can be mitigated by using optimisation methods. One of these optimisation methods consists of reducing the complexity and size of the data model used in the application. Another way to increase performance is to apply modern software design methodologies aimed at lowering software complexity and keeping the code up to date with the latest techniques.

An opportunity to combine DTs with web applications exists which will enable cross-platform access to DT functionalities.

1.1.4 Summary

DTs enable intuitive and interactive system representations. Web applications provide cross platform access to systems. Combining web applications with DTs may enable access for many users to DT functionalities.

DTs have been summarised into four components; firstly, the digital representation of the physical system, secondly, the sensor data, thirdly, the integration, computation, and analysis of the previous two, and finally, the 3D interactive model.

These four components have been grouped into three main research categories, namely, the data model, the reference architecture, and the user interface. Literature focusing on these three research categories will be studied and analysed in the rest of this chapter with a focus on the following proposed research question:

Can complex DT systems be optimised for use on mobile devices, such as smartphones?

1.2 Digital Twin data model design

1.2.1 Preamble

Complex engineering systems can be digitally defined in many different ways (e.g. using DTs). DTs are digital representations of physical objects with links to the physical object via, e.g. sensor data. For a DT system to be created, the data model that defines the DT must first be designed and created.

The data model of any system greatly influences the computing power required for that system to function. In this section, studies focusing on the design techniques of DT data models will be analysed.

1.2.2 Topological data models

There are two data model types which together form the greatest part of the DT data model, namely, the topological data model and the geometric data model [3]. The topology of a system describes the relation between components within the system, while the geometry describes the 3D geometric information of each component within the system.

The topology of a system can be seen as a higher level description of the system, while the geometry of a system is a much more detailed description of each component within the system. Both of these types of data are required for a DT data model. One needs to describe the geometry of each component within the system as well as their connectivity or relation to other components within the system.

In this section, studies focusing on topological data models will be analysed and in the following section, the same will be done for geometric data models.

Topological hidden facilities

Ortega et al. [3] chose the reference models used in CAD and Building Information Modelling (BIM) tools, as these tools are used to create 3D models which can be used in DTs. CAD data models contain the geometric data needed to represent the physical object in virtual space. Their approach in using CAD data models as a reference model is sound, as CAD tools have been the topic of research and practical applications for more than 30 years [12].

The data model from BIM tools can also be used as reference as it contains additional information such as sensor data linked to the component and documents concerning the component [3].

Ortega et al. altered these reference models using the requirements from the planned functionality for their topological DT software for hidden facilities such as sewerage systems. Their study focused on allowing users to inspect hidden facilities, e.g. underground pipes, and ventilation systems inside buildings using Augmented Reality (AR).

The system they designed also enabled the updating of nodes within the network via user input. Their study, however, did not focus on integrating sensor data into their model. Integrating sensor data into the model forms an integral part of the DT data model as it creates a virtual link between the physical and digital world.

Urban transport geographic information system

Zeng et al. [13] designed a data model for urban transport Geographic Information Systems (GIS). Their data model consisted of topological data (coordinates of each bus stop and the roads that connect them) and some additional information, e.g. vehicle capacity and speed.

The data model was also designed to contain route usage information, such as the average travel time between certain bus stops. This data model was used to generate a transport network for the analysis of existing public transit routes and to assist in improvements and additions to the transit network.

This data model was also designed to generate route usage information as transportation is progressing. In transit, the GIS used by Zeng et al. was used to track and update the route information. This real-world route generation is advantageous for large urban areas with many public transit services as all routes can be generated and analysed through one centralised database.

Real-world route generation provides many advantages for urban planners as it removes the manual creation of transit routes. The system created also enables urban planners to improve transit routes, minimise the time that commuters spend on public transit, and reduces carbon emissions. These are all valuable advantages which were enabled by their data model design.

The result from the study of Zeng et al. was, however, only single linear route generation. Their proposed solution was not verified using a complex system such as the compressed air network of a deep-level mine. These networks typically consist of hundreds, if not thousands, of connected tunnels with many sensors in each of these tunnels.

Multidimensional cadastral data model

Ding et al. [14] proposed a multidimensional cadastral data model which consists of four-dimensional components (a set of three data models over the change of time). These three models are the cadastral object (legal documentation regarding land ownership, etc.), the geometry component (which they define as the 3D geometric representation of the cadastral object), and the topological component (shared topological data which creates links between multiple cadastral objects).

This data model provides valuable information of the change of cadastral data over time. This can be useful for interested parties who, e.g. require information regarding cadastral trends in a province over a certain period of time for auditing purposes.

Ding et al. also described methods with which their data models can efficiently be queried from a database. This makes their reference model appealing, however, the scope of the current study does not include keeping track of the change in data over time which could cause unnecessary complexity in the data model.

This needs to be avoided as it increases the required computing performance [15]. Also, their data model did not account for information such as sensor data which forms an important part of the DT system as it is one of the links between the virtual representation of the system and the physical system.

Studies focusing on the design of topological data models have been analysed. Topological data is required in a DT as it describes how components within the system are connected to one another. The focus now shifts towards geometric data models.

1.2.3 Geometric data models extended with metadata

Geometric data models contain the geometry data (shape, size, rotation, etc.) for each component within a system. These data models typically contain more data than topological data models, as geometric data describes each component within the system in much more detail than topological data [16].

These geometric data models are also typically extended with metadata, e.g. time-series data gathered from sensors installed on components within the system and documents related to component installation or maintenance schedules [17].

Multi-scale geometric data

Borrmann et al. [16] proposed a geometric data modelling method specifically for shield tunnels which extends the Industry Foundation Classes (IFC) standard (which has since then been standardised in ISO 16739-1:2018). The IFC standard was developed to enable seamless data exchange in the infrastructure engineering and design industry and in this domain it has been widely used for the integration of simulation and analysis tools [18, 19].

Integration of simulation and analysis tools into existing systems provides many advantages for engineering design and optimisation. These tools perform calculations, given certain input parameters of a system, and produce an approximation of the system response which enables users to predict the performance of the system. These tools also assist users to test possible improvements to the performance and lifetime of a system.

The data model Borrmann et al. proposed was adapted from the IFC to support multi-

scale data. Multi-scale geometric data has the advantage of representing a system in varying proportions [16]. This is desirable in systems where there are large portions of sparse geometric data, i.e. geometries which simply connect two or more points of interest in the system, e.g. a pipe supplying compressed air to different stations within a deep-level mine.

Storing all the mesh data representing the surface of the pipe is unnecessary, as the pipe can be described simply by a set of coordinates of the route it follows and the diameter of the pipe. Storing only the required information of each component within a system will produce a data model which is significantly smaller than one that stores every geometric detail.

Information in a system that is deemed to be important can be determined by the requirements set out for the system, and these requirements can be derived from the purpose or desired function of the system [20]. The results obtained by Borrmann et al. showed that their approach produced the desired multi-scale geometric data with varying levels of detail. However, in order to use their data model, they developed a native application to execute on a desktop computer.

The proposed data model and case study by Borrmann et al. does not account for cross-platform access to the DT functionalities. They also did not analyse the performance of the application which was using the data model to represent the shield tunnel—their result was only that the system was operational. Their study did, however, focus on using a well-defined industry standard and Borrmann et al. proposed a data model to develop the standard further.

Data models for manufacturing applications

Zhao et al. [21] defined a DT process model for use in manufacturing applications such as machining. Their model was designed to represent single components for manufacturing. This data model, however, is complex as it contains as much relevant information regarding the component as possible. Their proposed model consists of five categories of data: simulation, perception, detection, entity, and process.

The simulation category consists of computer-generated predictive data regarding the attributes of the component after it has been manufactured (e.g. tensile strength). The perception category consists of component status information (e.g. the quality of machining and environmental impact).

The detection category consists of fault detection parameters in order to pre-emptively avoid manufacturing faults. The entity category describes the geometry of the component and machining tolerance. Finally, the process category describes the machining procedure and also contains information regarding the equipment to be used in the machining process.

This data model is complex as it contains information regarding not only the geometry of the component itself, but also information related to the machining procedure, fault detection, simulation data, etc. This data model was designed specifically for manufacturing applications, and can effectively be used in this context, as verified by Zhao et al. in their case study on a diesel engine connecting rod.

Their proposed data model is a valuable reference model for manufacturing applications as it contains a broad range of information regarding the manufacturing process. The fault detection category enables manufacturers to increase production efficiency, as faults can be pre-empted, avoided, and wastage can be reduced. The simulation category enables manufacturers to improve their component design through iterations, which results in higher quality products being produced.

Their data model, however, was not designed to contain large sets of coordinates and dimensions required in order to, e.g. model a deep-level mine which consists of thousands of different components. Their data model was also not designed to contain time-series data for many sensors present in deep-level mines.

Smart city modelling

Ruohomaki et al. [22] designed a platform to enable DT functionalities and BIM for cities. For their platform they used the CityGML standard data model. It is an extension of the Geography Markup Language (GML) which was created using the Open Geographic Consortium's abstract specification [23]. This specification models physical systems using features, which is a digital representation of the geometry and other real-world attributes of the objects within the system.

The CityGML data model, which Ruohomaki et al. used for their system, stores geometric, topological, semantic, and appearance data. One advantage of the CityGML data model is its modularity. It does not need to be redesigned in order to change or add any information if this information falls within one of the categories of data available within the CityGML model.

The platform created by Ruohomaki et al. was designed to model a city in terms of its geometry, topology, and other publicly available information (e.g. energy performance classifications for buildings). This platform was applied to a case study in the city of Helsinki for the Kalasatama district.

Their platform enabled city officials to plan for future improvements and maintenance in the district using the DT's 3D representation of the district as well as various metadata for the buildings within the district. It also enabled citizens to inspect the plans proposed by the officials for public comment.

Their data model, however, did not include the integration of sensor data. Their study also did not focus on cross-platform access to the DT functionalities. They did mention that further development to the platform was required as their method to create the platform did not include this. Their data model is considered scalable as they have used the CityGML data model within their platform, which is a highly modular and standardised data structure.

Digital Twin scope and requirements

Stark et al. [5] proposed a generic method to define the scope and requirements of DT systems. Their proposed method included considerations for the high-level definition of the scope and requirements, specifically for DT data models.

Many studies have done this for their specific applications, however, few studies have been found that create such a generic method to define the scope and requirements. The abstractions Stark et al. used to represent the process can be applied to many different applications and industries.

Their proposed method consists of using the following categories (including examples) which must be defined before a DT system and data model should be designed or extended:

1. integration breadth: interconnection of components within the system;
2. connectivity: flow of data between the physical and virtual systems (unidirectional, bidirectional);
3. update frequency: near real-time or historical data;
4. Cyber-Physical System (CPS) intelligence: autonomous control or human triggered strategies;
5. simulation capabilities: predictive maintenance and optimisation;
6. data model features: system geometry, topology, and metadata (e.g. sensor data);
7. human interaction: Human-Machine Interface (HMI) such as AR; and
8. product lifecycle: design phase, maintenance phase, and end of life.

Although they did not go into much detail regarding the data model structure, their method does include consideration for the integration of sensor data into the DT data model. Their case study provides a practical example of their proposed method for a DT system representing a smart factory cell.

The study by Stark et al. has been included in this section due to their approach of proposing a generic method to define the scope and requirements of DT systems and more specifically, DT data models.

Their proposed method, however, does not focus on designing the data model for low-powered devices, such as mobile devices, to access DT functionalities. Their study also did not consider scalable cross-platform access to DT functionalities.

1.2.4 Summary of Digital Twin data model design

Studies focusing on the process of DT data model design were investigated and analysed. It was found that DT data models typically contain topological data, geometric data, and metadata such as sensor data used to analyse the performance of a component or system [3].

Table 1.1 summarises the focus areas of each study analysed. Almost all of the studies focused on reference architectures, while only three of the studies focused on sensor data integration into the data model. Two of the studies focused on cross-platform access and two studies focused on scalable functionalities. Only one focused on continuous development and maintenance, while not addressing any of the other categories.

There is a clear gap for a DT data modelling method which focuses on sensor data integration, cross-platform access, scalable DT functionalities, and continuous development and maintenance.

Table 1.1: Summary of literature analysis on DT data model design

	DT reference architecture	**Sensor data integration**	**Cross-platform access**	**Scalable DT functionalities**	**Continuous development and maintenance**
[3]	■		■		
[13]	■	■			
[14]	■		■	■	
[16]					■
[21]	■	■			
[22]	■			■	
[5]	■	■			

■	Within the scope of the study
	Not within the scope of the study

DT data models, which is the first research category shown in Figure 1.3, have been investigated and studies relating to it have been analysed. In the following section, the second category shown in the figure will be investigated, namely DT reference architectures.

1.3 Digital Twin reference architectures

1.3.1 Preamble

Complex systems contain many different software applications and databases which need to communicate in order to achieve a certain goal. These complex systems may, however, run into issues (e.g. scalability problems) if the system reference architecture was not designed to achieve certain requirements [7]. Studies focusing on the design of DT reference architectures are analysed in this section.

1.3.2 Application-specific reference architectures

The architecture design of a system has a significant influence on the performance of the system, e.g. it determines to a large extent the scalability of the system [2]. The platforms for which these systems are created and the types of backend computing used significantly influences the scalability of the system. The current study is specifically investigating highly-scalable cross-platform systems for DTs.

Architecture designs which include integration with cloud computing are much easier to scale than, e.g. systems using an on-site processing unit [24]. The architecture design should therefore form a critical part of a DT system framework.

Health monitoring using smart devices

Laamarti et al. [25] proposed a new ISO standard for a DT framework for health monitoring using smart devices. This framework consists of data sensing, data collection, data analysis, and DT interaction.

Data sensing in their proposed architecture consists of health monitoring devices which are found in, e.g. smart watches. Data collection consists of receiving the sensor data from the health monitoring devices via an Application Programming Interface (API) and the storage of this data into databases.

The data analysis consists of cloud computing tools to identify patterns and perform algorithms on the collected data. Lastly, the DT interaction consists of a mobile application for the user to

connect their health monitoring devices to the system and to inspect the information generated from the system after it has gathered sensor data.

Laamarti et al. did integrate live sensor data and cloud computing into the proposed architecture for health monitoring. However, the architecture did not include a geometric model of the system that was analysed (i.e. the person wearing the devices). The data was simply shown in graphs and tables on the mobile application interface. This method is sufficient when there are not many sensors within the system. However, once many graphs containing each sensor's name are shown, then the physical context is not as apparent when compared to a 3D model of the system.

The proposed architecture by Laamarti et al. is scalable in terms of data processing and storage as these two functions were implemented using an API and cloud computing. An API increases system scalability as its function is to standardise communication between integrated systems [26]. Utilising an API means that each system does not need to take into account every other integrated system. The communication in each system simply needs to be implemented by following the standards set by the API and then the system can communicate to any other integrated system.

Their architecture, however, did not include a 3D representation of the system being analysed and is hence considered not scalable in the context of this study. The DT interaction was also only available on a mobile application and not any other platform. Their study also did not consider continuous development and maintenance in the DT system.

Mine ventilation control system

Kychkin and Nikolaev [27] proposed an architecture for an Internet of Things (IoT)-based mine ventilation control system. Their architecture consisted of four subsystems, namely the HMI, the DT, the Information and Communications Technology infrastructure, and finally the physical object subsystem.

Their study did not discuss the reference architecture in detail, as their main focus was to calculate the energy forecasting for the ventilation control system within a mine. However, their reference architecture did include a web-based HMI, which will enable cross-platform access.

For their reference architecture they proposed cloud computing for the backend, which enables high scalability. A DT system with cross-platform access combined with the scalability of cloud computing will significantly increase the number of users that have access to DT functionalities, when compared to a DT application which requires a desktop computer and local server to

execute.

The proposed architecture did not include considerations for a 3D model for user interaction. A 3D representation of the system is one of the many advantages of using a DT to model the system.

Although the main focus of their study was not the DT reference architecture, their study was one of the few found that proposed the use of a web application as the interface to the DT system. They also verified their architecture by displaying the calculated energy forecast for their case study on the web application. For this reason, their study was included in this section.

Thermal comfort assessment

Shahinmoghadam et al. [20] proposed an architecture to monitor near real-time thermal comfort conditions and is shown in Figure 1.4. Their architecture was built using three layers, namely a data acquisition layer, a middleware layer, and an application layer.

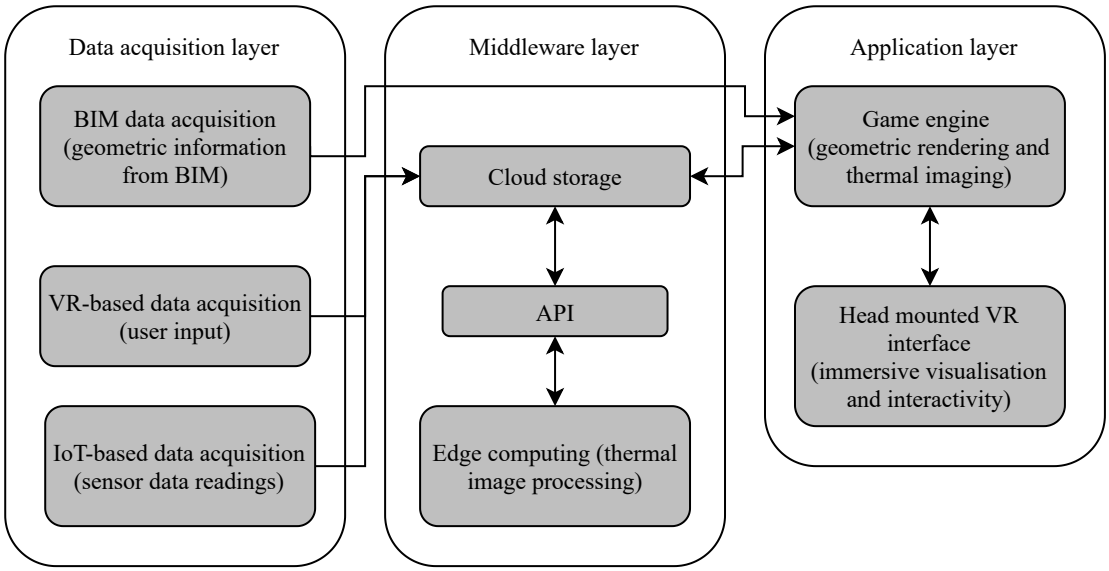


Figure 1.4: Thermal comfort assessment architecture. Adapted from Shahinmoghadam et al. [20].

The data acquisition layer consists of various systems through which data is collected for the environment to be assessed. These systems include BIM (through extracting geometries from the model), Virtual Reality (VR) (collection of user-input via the interface), and IoT (output from various thermal sensors throughout the environment).

The middleware layer consists of cloud storage, an API, and edge computing. These three systems collect and store the data obtained from the data acquisition layer, perform calculations on the data (such as the processing of thermal imaging), and parse user-collected data into the

geometry of the environment. The application layer consists of a game engine on which the VR system is executed and a head-mounted display for the VR HMI. The VR system displays near real-time sensor data as well as the parsed thermal images of the environment.

Using their architecture, they achieved the goal set, which was to create an interactive user experience in VR which shows near real-time sensor data as well as the output graphs from calculations in cloud computing. Their architecture is however, limited by the game engine and hardware necessary to enable VR interaction. These game engines are native applications and typically require powerful desktop computers to execute[3] and will therefore not be suitable for this study, as it does not enable cross-platform access.

Although their user interface is more immersive than, e.g. interacting with a computer screen via mouse and keyboard, their model consists of only one room. The reason is that with a VR headset the user's range of movement is extremely limited, as the headset is often connected via cables to a computer executing the game engine. Their architecture will not be suitable for this study as it was not designed for large and complex systems.

Cyber-Physical Production Systems

Ashtari et al. [28] proposed two architectures for DT applications in Cyber-Physical Production Systems (CPPS). The difference between the two architectures being that the latter includes artificial intelligence capabilities. Only the first proposed architecture will be considered as artificial intelligence is considered out of scope for this study. Their proposed architecture consists of the following four tools and systems:

- Storage and interactive tools: enabling persistence of the data model and user interaction,
- A data acquisition interface: for the collection of operational data (e.g. from sensors),
- A co-simulation interface: enabling simulations over multiple DTs to increase scope, and
- A synchronisation interface: in order to keep the DT aligned with the physical system.

They applied this architecture to an experimental prototype with a specific focus on the synchronisation of DT data models in a modular production system. The results obtained from the case study showed that changes made to the physical system was detected by their synchronisation system and notice of updates were issued in the DT models.

This synchronisation tool is advantageous because in order to utilise the full potential of a DT, it must reflect the real-world system as accurately as possible. As real-world systems tend to change frequently, it is necessary to also frequently update the DT. Automatic notice of the

[3] "Hardware and Software Specifications," Unreal Engine. Accessed on: May 10, 2021. [Online]. Available: https://docs.unrealengine.com/4.26/en-US/Basics/RecommendedSpecifications/

requirement to update the DT reduces the possibility that it will fall out of synchronisation with the physical system.

Ashtari et al. also applied their co-simulation concept to a hypothetical scenario which consisted of a smart warehouse with temperature control. The simulations were performed in MATLAB/Simulink, OpenModelica, and Java. Their results showed that the output from each simulation was re-usable in the other simulations that ran concurrently with it.

Their strategy enables the detailed simulation of modular systems, which provides low-level performance analysis. It also enables the input from and output to other simulations, which provides a higher level overview of the performance of the entire integrated system. Both of these are highly desirable as each system can be analysed separately, in combination with other systems, and can be analysed as a whole.

Microservices for smart manufaturing

Damjanovic-Behrendt et al. [29] proposed a microservices architecture for a DT demonstrator in the manufacturing industry which they applied to a case study using open-source tools. Microservices are applications that are built up of loosely-coupled services, each of which have a single and well-defined purpose, e.g. a data acquisition service.

Building applications in this way enables the application to be highly scalable, as these applications can be hosted on cloud computing services [29]. These cloud computing services enable up or down scaling of services as required with ease.

The architecture proposed by Damjanovic-Behrendt et al. consisted of five components as seen in Figure 1.5, namely the virtualisation manager, the data manager, the model manager, the service manager, and the interoperability component.

The virtualisation manager component consists of the following microservices: (1) the front-end services which consisted of the graphical user interfaces (GUI) to interact with the systems, (2) simulation services for the manufacturing process, (3) monitoring services of the manufacturing process such as dashboards and alerts, (4) events management services such as critical event detection which enable users to mitigate their effect, (5) simulation management services which gather data from manufacturing process in order to improve simulation accuracy, and lastly, (6) decision-making and control services for conflict resolution and control over DT functionalities.

The data management component consists of the data acquisition service (gathers all required data from the manufacturing process and sensors within the systems), the knowledge discovery

service (analyses received data through calculations in order to send instructions to the control system), and the data analytics service (machine learning services).

The model management component consists of data computing and data representation services which enables the configuration and analysis of a DT manufacturing system model. Damjanovic-Behrendt et al. did not focus on the detail of the initial setup or configuration of the DT model, they rather focused on the overall system architecture.

The service management component enables and maintains the connectivity between services and the manufacturing system such as sensors, actuators, etc. It also consists of cybersecurity services, to ensure high levels of security between and inside each of the services within the architecture. The notebook analytics services, which also form part of this component, enable state save mechanisms which can share system configurations and performance information with other DT systems.

Finally, the interoperability component consists of integrating with measuring devices within a system (e.g. sensors) in order for the DT system to receive input from the system and to enable the DT simulation functionality.

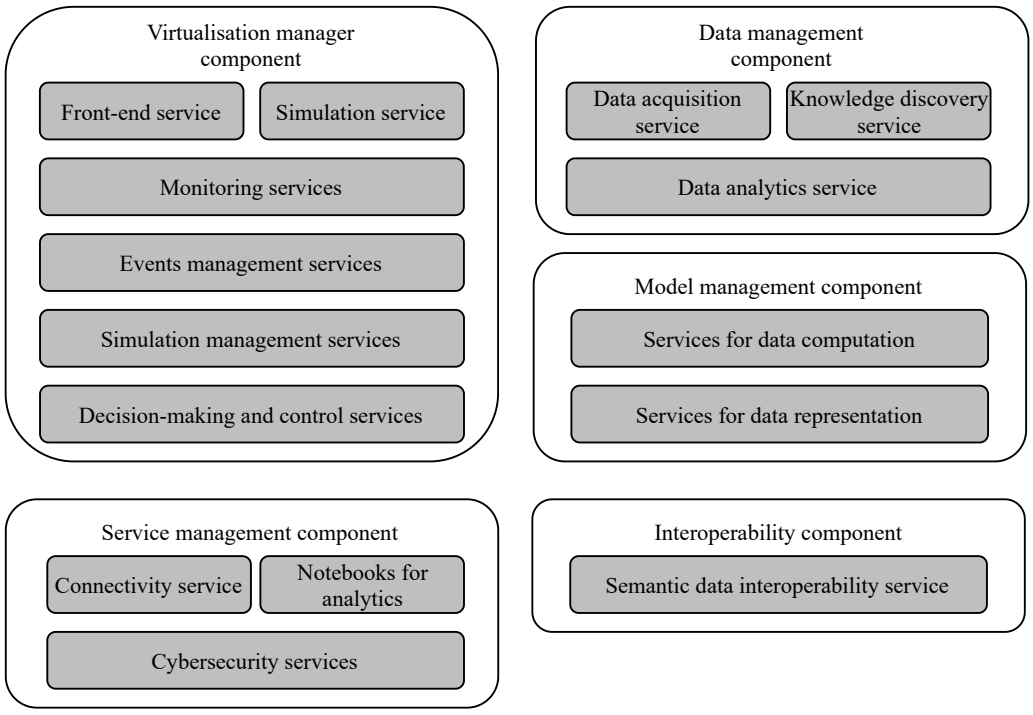


Figure 1.5: Smart manufacturing DT microservices architecture. Adapted from Damjanovic-Behrendt et al. [29].

Damjanovic-Behrendt et al. applied this architecture using open-source software on a case study which consisted of a DT demonstrator for a smart manufacturing system. While they only proposed a possible solution for the case study, they did not verify the solution by implementing and testing it. That being said, their proposed architecture has still proven valuable as few studies have been found that propose a DT architecture which consists only of microservices.

Some architecture designs are tailored to specific applications which may lead to improved performance for that application, however, this approach cannot be used to provide a reference architecture which applies in many different applications. Some design techniques within these application-specific architectures may be of use when designing a generic architecture [30].

Studies relating to application-specific reference architectures have been analysed in this section. In the following section, generic DT reference architectures will be investigated and studies relating to it will be analysed.

1.3.3 Generic reference architectures

A generic reference architecture has increased levels of abstraction when compared to an application-specific reference architecture. This means the generic architecture does not contain specific implementation details for each component within the architecture, but rather provides high level guidelines to achieve a certain goal.

Generic reference architectures do, in fact, have the advantage of being usable for many different applications in many different industries. Due to this, many more system designers can use and possibly improve or extend the generic reference architectures.

Sustainable CPS architecture

Broo and Schooling [30] defined a Cyber-Physical System (CPS) as a system which binds physical processes with computing processes. They also found that most studies regarding CPS focused on manufacturing applications. They proposed a generalised CPS architecture which consisted of four layers, namely:

1. the goals the system should achieve (e.g. carbon neutrality and sustainable development),
2. the stakeholders interested in achieving the goal (e.g. industry representatives and citizens),
3. the integrated system-of-systems built to achieve the goal (DTs of buildings, roadways, railways, power grids and also data acquisition and storage systems), and
4. the services the system provides (e.g. energy management and logistic management).

Broo and Schooling did not go into detail regarding DTs (third layer) or the scalability of the architecture. They rather focused on creating a generalised CPS architecture for multi-stakeholder involvement.

Their proposed architecture and their discussion thereof provides context regarding where DTs could typically be used in large complex systems such as smart infrastructure systems. They also discuss strategies, the most probable shortcomings which these types of systems can encounter, and possible solutions to them. Broo and Schooling provide a strategic resource which is valuable when CPS are in the planning and design phase.

Cloud-based CPS architecture

Alam et al. [24] proposed an extension to the CPS. They altered and extended the CPS to focus specifically on integrating the DT with cloud-based computing; cloud-based CPS (C2PS).

In their reference model every physical object is represented as a DT with physical sensors. The reference model was designed to handle many small DTs which can communicate between each other and a central database, which is similar to the IoT concept. Their reference model assumed an open network structure between all the DT entities.

The study done by Alam et al. did not focus on using these DT entities as a 3D visualisation tool of data and information for users. They rather focused on the possibilities that exist when cloud-based DTs (which model single components with sensors) can communicate with one another and a central database to create information (such as the output of calculations) and then to present the output to the user.

Their micro-DT approach is appropriate when data is gathered from the components within a system. However, in the case of the current study, the DT also needs to be used to create an interactive 3D model for the user of the entire system. Hence, splitting the system into single component DTs will add unnecessary complexity.

Digital Twin as a service architecture

As previously mentioned, most studies focused on DTs consider only the manufacturing applications. However, Aheleroff et al. [2] proposed an architecture reference model for DTs for use in a wide variety of engineering industries: a DT as a service reference model for scalable individualisation to be achieved.

Individualisation refers to the personalisation of software for each user. This personalisation may be unique interfaces or custom features that the user requires, but developing software specifically for each possible variation of a system based on all user's preference is also not sustainable [2]. Hence, Aheleroff et al. proposed a modular architecture, which enables sustainable individualisation of systems using DTs.

Aheleroff et al. applied their architecture to a case study which consisted of real-time data collection, remote control, and cross-platform access. They implemented an AR mobile application to monitor wetland data. The AR application could overlay real-time data and supported user interaction with the model.

This AR application could be expanded to integrate GIS to show a user digital data in real-world physical relation to their position. However, the models shown on the mobile device were not complex models of thousands of connections and did not have as many data collection points. Therefore, the use of their proposed architecture for cross-platform access has not been verified for complex systems which require high scalability.

Architecture for degradation evaluation

D'Amico et al. [4] proposed a framework for the evaluation of degradation within complex engineering systems, shown in Figure 1.6. The overall system for which the DT was created has been divided into sub models for the degradation of each component to be evaluated separately.

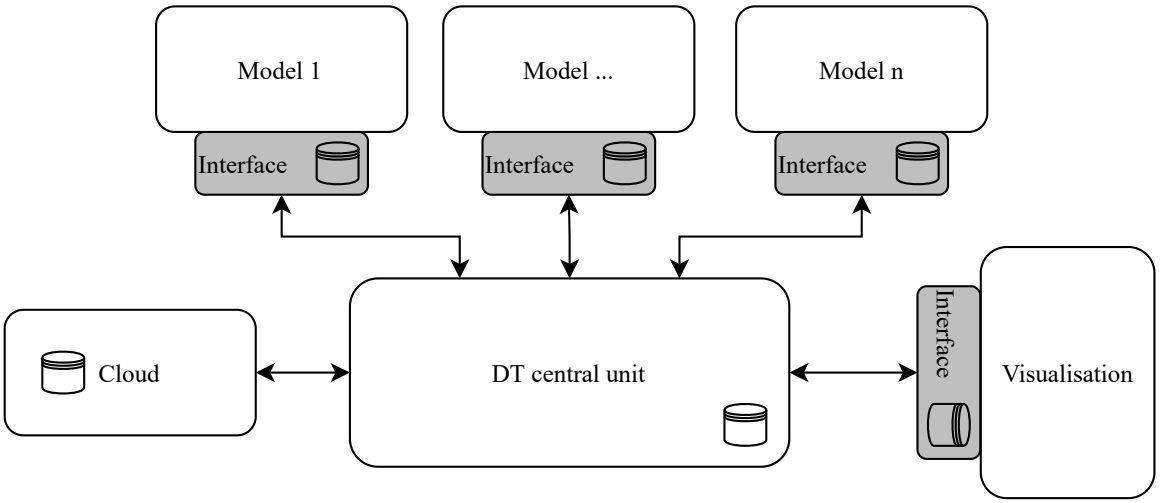


Figure 1.6: Conceptual framework of the DT. Adapted from D'Amico et al. [4].

D'Amico et al. discussed scalable asynchronous communications between the central unit and the various interfaces. The framework also includes cloud storage and computing, which both enable scalability of the framework. Their degradation assessment model takes, as input, the geometric data of the component as well as near real-time sensor data.

Their study did not focus on the visualisation of the DT; however, they did comment that an interactive 3D model can be implemented using a web application or AR.

Generic architecture for energy systems

Steindl et al. [31] created a Generic Digital Twin Architecture (GDTA) model for industrial energy systems, as shown in Figure 1.7. Their study aimed to unify many previously proposed application-specific architectures into one generic DT architecture. They mainly considered studies from the manufacturing and process engineering fields.

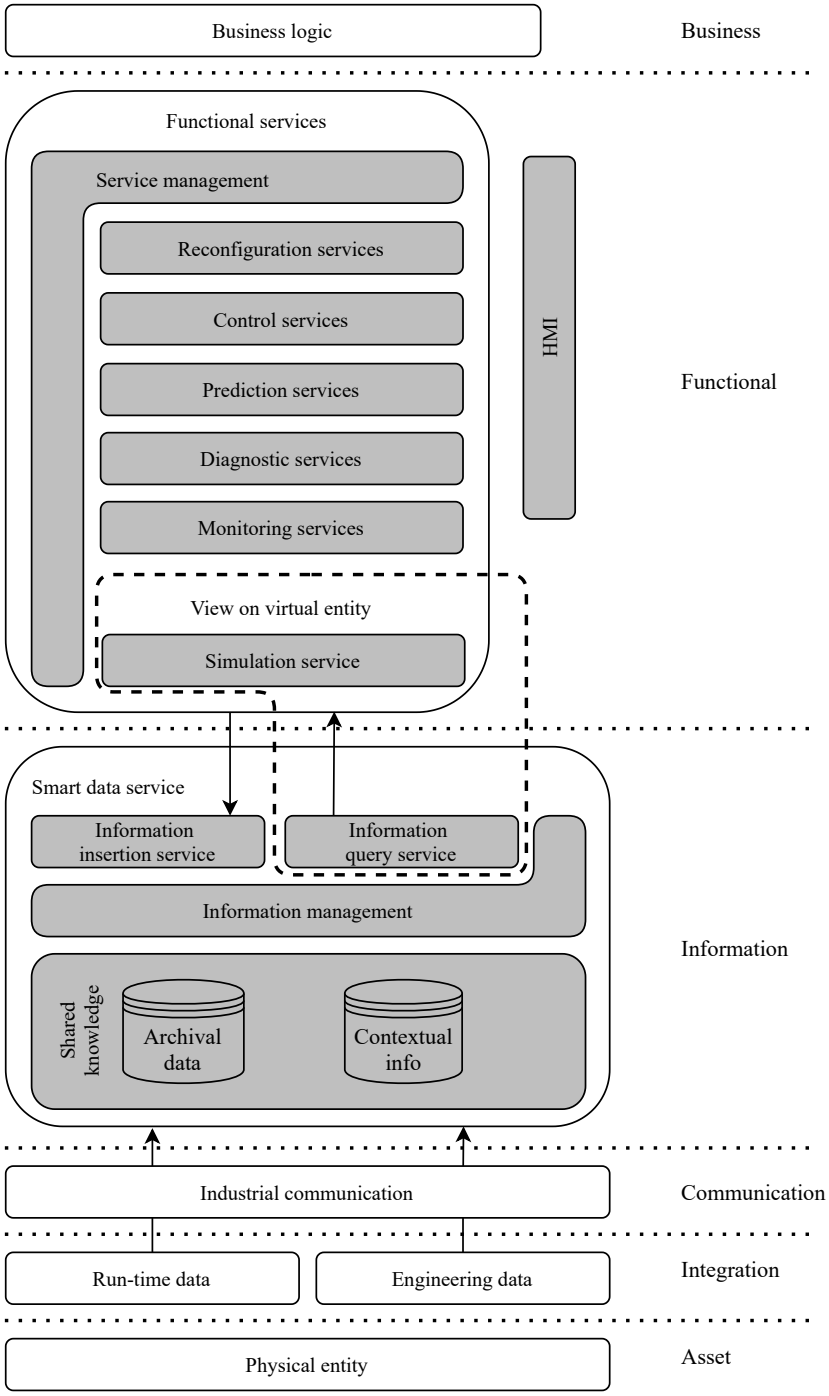


Figure 1.7: GDTA model. Adapted from Steindl et al. [31].

A DT entity is not defined within it as the model should be usable in many different applications where the DT is defined in many different layers. Therefore, making it a generic model.

Their research included a case study in which a thermal energy storage device was simulated using a system designed with their GDTA model. The case study was a relatively small system, as it consisted of one component and multiple pipes. The simulation software was used to predict the change in output temperature of the thermal storage device for a specific load cycle.

Steindl et al. did discuss cloud computing and APIs for system scalability, however, it was not within the scope of the study to optimise the model for low-powered mobile devices. The study also did not focus on cross-platform implementation of the HMI. However, the GDTA model remains a valuable reference architecture for many different applications not only in the field of industrial energy systems to implement DT systems.

1.3.4 Summary of Digital Twin reference architectures

Table 1.2 summarises the results of the analyses of studies which focused on DT architecture reference designs. All the studies also focused on sensor data integration. However, only two studies focused on cross-platform access in the architecture design and only one focused on scalable DT functionalities, and continuous development and maintenance.

Table 1.2: Summary of literature analysis on DT reference architectures

DT reference architecture	Sensor data integration	Cross-platform access	Scalable DT functionalities	Continuous development and maintenance
[25]	■			
[27]	■	■		
[20]	■			
[28]	■			
[29]	■		■	
[30]	■			
[24]	■			
[2]	■	■		
[4]	■			
[31]	■			

■	Within the scope of the study
	Not within the scope of the study

It is clear from Table 1.2 that there is a gap in DT-related research in terms of the combination of cross-platform access to DT functionalities, scalable DT functionalities for complex systems,

and continuous development and maintenance on the DT system. This correlates with the conclusion made from Table 1.1, that there is a clear gap for a DT reference architecture which includes cross-platform access, scalable DT functionalities, and continuous development and maintenance.

DT data models and DT reference architectures, the first two research categories shown in Figure 1.3, have been investigated and studies relating to them have been analysed. In the next section, the third and last category identified will be investigated, namely DT applications.

1.4 Digital Twin applications

1.4.1 Preamble

Studies focusing on DT data models and reference architectures have been analysed and a possible gap has been found for a study focusing on cross-platform, scalable access to DT functionalities with a focus on continuous development and maintenance.

Now, the topic of the literature analysis moves on to the applications of DTs and what benefits and opportunities they may provide.

1.4.2 Industrial applications

Industrial systems consist of large collections of sub-systems which have been integrated to achieve a set goal (e.g. a machining system which shapes a material into the required geometry). These industrial systems typically contain many different integrated components and sensors which need to be monitored to detect any possible faults and also to enable technical personnel to determine optimisation opportunities [21].

Role of the Digital Twin in smart manufacturing

Tao and Zhang [6] discussed the role that the DT will have in the era of smart manufacturing. They proposed an operations mechanism for a DT shop floor which included the steps required before, during, and after production. Steps were also included in which the component to be produced as well as the production process are to be simulated, evaluated, and optimised.

Their proposed mechanism accounted for the entire process of production in order to maximise the benefit that the DT system would provide. Some of the categories included in the mechanism before production are production plans, resource allocation services, data integration services (sensor and model data), and simulation services for the production plan.

Next, the system must be evaluated and models should be calibrated using the real-world data collected during production. This calibration aligns the DT system with the physical system. This calibrated DT model is then to be used for optimisation strategies. This enables technical personnel to increase system efficiency as most of the information present in the physical system has been consolidated into the DT.

These simulations provide valuable insight into the expected performance of the system as well as provides the opportunity to test possible improvements to the system. Their study proposed models for data integration between the physical space (the manufacturing components and sensors) and the virtual space (data storage, cloud-computing, and simulation).

However, they did not apply their mechanism to a case study to evaluate performance. They also did not cater for a 3D model representing the system for a possible HMI, they only mentioned that graphs would be used for final system output.

CPS for autonomous manufacturing

Ding et al. [32] proposed a DT-based CPS for autonomous manufacturing. They discussed the value of integrating real-time data, simulations, and predictions to improve production system performance. They also discussed that sensors and embedded devices have become much more affordable and readily available than a decade ago. Because of this, real-time data collection does not typically cause feasibility issues when certain components within the system require sensors.

The advantages of being able to collect real-time data and store it for historical analysis far outweigh the cost of procuring and installing these devices [32]. Some of these advantages include the possibility for automatic or manual fault detection through a DT interface which has been integrated with the sensor data.

Ding et al. also proposed an operating flow methodology for autonomous control and fault detection in which the collected sensor data formed a feedback loop from the physical system to the DT system. However, their proposed system may also be a valuable tool for technical personnel to monitor performance and it may enable the identification of improvements to the production system and process.

Potential for Digital Twin systems in product design

Wagner et al. [1] focused on the applications of DT systems in product design within Industry 4.0. They discussed that DTs can be utilised for any part of the product life cycle,

namely the product design phase, the product creation or manufacturing phase, and the product use phase.

During the design phase, Wagner et al. discussed that DTs can be used to model the product to be manufactured. A significant advantage of this is that the product design can be based off of previous geometric models and alterations can be made to improve the product based on feedback. This increases the time- and cost-effectiveness of product design.

During the manufacturing phase DT systems can be used to model the manufacturing system or process. Doing this enables real-time monitoring of system performance, creates simulation opportunities to optimise the process, and provides possibilities for autonomous fault detection during the manufacturing process [6, 21, 28, 29, 32]. All of these advantages increase system time- and cost-efficiency as well as product quality.

Finally, Wagner et al. also discussed the possibilities for DTs in the product use stage, such as degradation evaluation within complex engineering systems [4]. This creates opportunities for real-time system monitoring, simulations of system performance to enable optimisation, and autonomous control. As discussed by Wagner et al., there are many other advantages that these DT systems may provide in the manufacturing industry which has not been studied yet.

Robotic Digital Twin system

Kaigom and Roßmann [33] applied DT techniques to complex robotics applications. They proposed and implemented a system for torque-based robotic arm control. Their system included services for the simulation of torque experienced by the different joints within the arm.

Their system also included physical position control and visualisation services for the robotic arm. The physical position control was implemented for automatic and manual control. This enables operators to control the robotic arm remotely which is advantageous in dangerous environments such as in outer space or during the process of storing hazardous products [34].

The automatic control also minimises the human-error in, e.g. manufacturing as the arm has a predefined and simulated movement pattern which should not change over time, whereas human beings experience fatigue and make mistakes.

Their proposed system was based on a DT reference architecture, which increases reliability in their proposed system as it was extended from other peer reviewed studies. Their proposed system also includes sensor data integration (e.g. torque sensor data) as well as an interface to visualise the DT as control and simulation is in progress.

They did, however, implement their system through a native application for desktop computers. Therefore, cross-platform access to the DT functionalities was not achieved. Their study is still a valuable resource for robotic applications requiring simulation and automatic control (which increase system efficiency and minimises human error).

Digital Twin for cooling towers

Blume et al. [35] proposed a workflow to create DTs for building services operation such as cooling towers. The purpose of their study was to increase system understanding and performance simulation and analysis. To do this, they proposed the aforementioned workflow.

The workflow consisted of three main phases, namely the development of the business understanding, the data modelling, and the evaluation phase. The business understanding phase consisted of analysing the requirements from the service the building provides where the cooling tower was required to produce a drop in the water temperature to a certain level. They discussed that other characteristics of the service may also be analysed. This provides the context in which the DT can be created.

The data modelling phase consists of three categories, namely the data understanding, the data preparation, and the final modelling step. These three categories encompass the process of analysing the data present in the system, preparing the data for the modelling process, and finally, creating the data model from the analysis.

They used an artificial neural network to extract non-linear relationships between sub-systems of the service in order to use in the evaluation phase. The result from this analysis may provide insight into the service operation and bring to light improvements to be made to the system. Artificial neural networks are considered out of scope for the current study and will not be analysed further.

The final phase, the evaluation phase, consists of taking the results from the data modelling process and communicating to the technical personnel the possible improvements to the system. These improvements may increase the cost-efficiency and performance of the cooling tower.

Blume et al. implemented their case study on a desktop computer using a native application, hence, the system is not cross-platform. Their workflow also did not contain any reference to this aspect. Their workflow did account for sensor data, however, they did not discuss the scalability of the system nor whether or not continuous development would be performed.

Industrial applications have been investigated and it has been identified that DT systems

provide many valuable advantages, such as autonomous control and automatic fault detection. Many of the studies investigated in the previous sections include discussions regarding simulation applications. Hence, in the next subsection, simulation applications are investigated.

1.4.3 Simulation applications

Many studies analysed have utilised DTs through simulation of the system for which the DT was created. Simulations provide valuable insight into the performance of a system, and when calibrated to the real-world system, can enable the identification of improvements through testing and analysis.

These simulations can be created during any phase of the system life cycle (design, manufacturing, and operation). During the design phase simulations can be used to, e.g. analyse product characteristics in order to align with design requirements [29].

During the manufacturing phase, simulations provide the opportunity for increased production efficiency [28]. During the operation phase, simulations enable the identification of further system cost and time efficiency improvements [31].

Digital Twin simulations for complex technical systems

Schluse and Rossmann [36] proposed the central use of DT data models in simulation-based development processes. The advantages of simulations has been discussed. Schluse and Rossman proposed an architecture for simulation services and used it in four applications.

The four applications were the servicing of spacecraft while in orbit, the localisation of machines used in the forestry industry, a driver assistance system, and lastly industrial system automation. The spacecraft service application was created to simulate docking and robotic arm control. This provides the opportunity for system designers to validate and improve their designs through simulations. This also creates training opportunities for personnel who need to operate the machinery.

The purpose of the localisation application was to enable the measuring of forestry inventory, i.e. the number of trees in the vicinity of the machine, in order to automate the inventory management and increase reliability. Automating this process removes the requirement for a person to do the inventory management which reduces the chance of human errors during the process [37]. The application further adds value by enabling the live tracking of machinery in the forest for fleet management and scheduling.

The driver assistance system consisted of simulating a roadway with cars driving in order

to simulate lane keeping and parking assistance systems. This was done by also simulating optical sensor readings which were attached to the virtual cars. These systems assist drivers by lowering risk while driving and reducing accidents when parking.

The final application was industrial automation. As discussed in previous sections, manufacturing and process automation systems using DTs have many advantages (e.g. increased time- and cost-efficiency). Schluse and Rossman, however, did not discuss cross-platform access, continuous development and maintenance, or address scalability considerations in their study.

Maintenance optimisation for multi-unit systems

Savolainen and Urbani [38] proposed a maintenance optimisation strategy for multi-unit systems. They split the strategy into two main components, namely the high-fidelity maintenance model and the low fidelity profitability model.

The maintenance model was based on established architectures which they combined into a model which consisted of analysing the time to failure of each unit in the system as well as analysing the time to repair by certain numerical distributions. The time to repair was specifically used to simulate preventative maintenance instead of waiting for unit failure before maintenance was done.

The profitability model focused on the throughput versus cost (i.e. cash flow) of running the entire system. This model consisted of a managerial cash flow model in order to estimate the sustainability of the system in different scenario's.

These two models were combined and applied to a case study which consisted of a mining industry fleet management system. Their application was aimed at optimising the high-level configuration of the mine. The mining operation consisted of a truck moving ore and other debris between a robotic shovel site, a dump site, and a workshop site.

Savolainen and Urbani simulated six different configurations of the mine layout by varying the number of shovels and workshops. Through simulation of the cost of maintenance and system performance, they determined the optimal throughput versus cost configuration for the mine. This showed that their proposed simulation model can increase the operational efficiency of industrial systems which is a desirable outcome in almost all industries.

They did, however, require a powerful desktop computer to execute the optimisations (which took an average of thirty minutes) [38]. Hence, their proposed solution will not be applicable for a cross-platform DT system. Their models were also not designed to integrate sensor data.

Asset administration shell using Digital Twin

Park et al. [17] proposed the Virtual Representation for DTs (VREDI) for use in an Asset Administration Shell (AAS). They described AAS as industrial systems which aim to enable the sharing of asset data between assets in a complex system and also the sharing of data with a centralised system for planning, scheduling, and condition monitoring.

Their first case study focused on the advantages of using an AAS in industrial applications. As this is not the focus of this study, the first case study will not be analysed. Their second case study focused on the implementation of the DT system for simulation, monitoring, and control of a complex industrial application.

The proposed VREDI application included simulation tools which enabled scheduling and layout optimisation in their case study. Both of these will enable the increase of system efficiency before the system has even been physically assembled. This will significantly increase the cost-effectiveness of the system.

VREDI also included visualisation tools for the 3D representation of the complex system. Their case study consisted of a complex factory logistics system, which included additive manufacturing, polishing, inspection, and the packaging of products.

Their proposed solution was applied to a complex system with many components working together to achieve a set goal. This, and their design philosophy of the DT system, shows that their system is scalable which means that their system will be able to handle even more complex systems.

Their system was, however, implemented on a native application for a desktop computer and, therefore, does not provide cross-platform access to DT functionalities.

A reference for Digital Twin applications

Qi et al. [39] analysed the DT model proposed by Tao et al. [40] (shown in Figure 1.8) and using the model as context, summarised frequently used technologies and tools for DTs in order to create a reference for DT applications in the future.

The model in Figure 1.8, proposed by Tao et al., consists of four components. The physical entity is the real-world object, process, system, system of systems, or organisation that the DT will be representing.

The virtual model is the representation of the geometry, properties, and rules of the physical

entity in digital domain. The services component consists of services which are enabled by the DT approach, e.g. simulation services, autonomous fault detection services [32], and maintenance planning services.

The DT data is the data generated from the physical entity (e.g. collected sensor data), the data generated from the virtual model (e.g. simulation data), and finally, the data generated from the services (e.g. configurations which notify technical personnel when data collected from a sensor exceeds certain set values).

Each of the four components of the model are interconnected which leads to one of the greatest advantages of DT systems; the fact that DT systems mirror physical systems in the digital domain and when metadata is shared between the physical and digital domain opportunities such as condition monitoring, predictive maintenance, autonomous fault detection, and simulation are enabled.

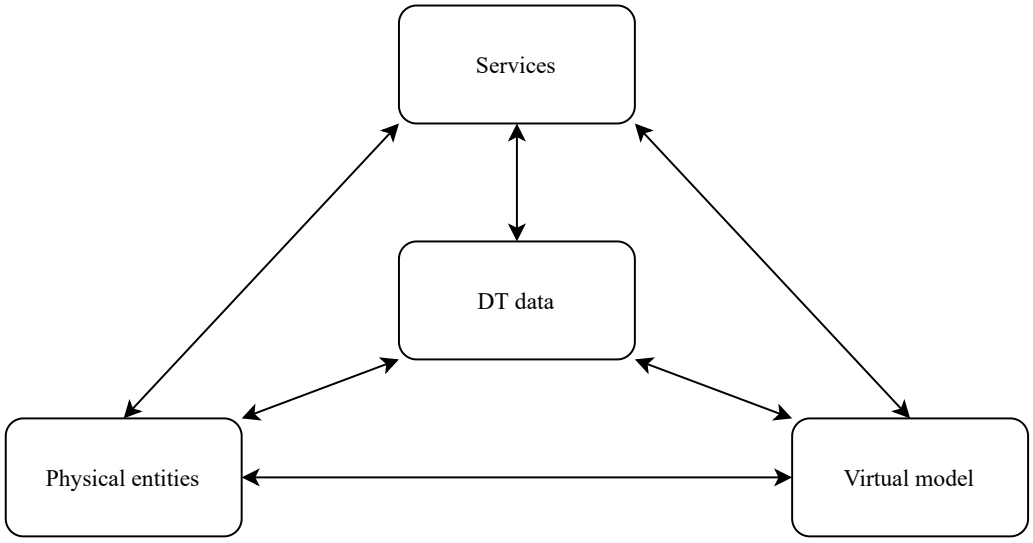


Figure 1.8: Generic DT model for applications. Adapted from Tao et al. [40].

Qi et al. further discussed the various fields and enabling technologies for DTs. The fields include manufacturing, aerospace, city, and healthcare. There have been studies that have proposed and implemented DT systems for some of these fields [21, 22, 25]. The enabling technologies include visualisation, simulation, modelling, and interface technology. There have also been studies that have proposed and implemented DT systems for these technologies [1, 6, 32].

1.4.4 Summary of Digital Twin applications

Studies focusing on DT applications in the manufacturing industry have been analysed. Some of these studies focused on using the DT to improve the design and simulation process of manufactured components, while others focused on the entire manufacturing process—from component design to production line optimisation. Others focused on preventative

maintenance.

Some studies have also been analysed which applied DT techniques to complex robotics applications [33] and wetland maintenance scheduling [2]. The analysed studies and their areas of focus are summarised in Table 1.3.

It can be seen that all but one study focused on sensor data integration, five included DT reference architectures in their methods, and four focused on scalable DT functionalities. Of note is that none of the studies focused on cross-platform access or continuous development and maintenance.

Table 1.3: Summary of literature analysis on DT applications

	DT reference architecture	**Sensor data integration**	**Cross-platform access**	**Scalable DT functionalities**	**Continuous development and maintenance**
[6]		■			
[32]	■	■		■	
[1]		■		■	
[33]	■	■			
[35]		■			
[36]	■	■			
[38]	■				
[17]	■	■		■	
[39]		■		■	

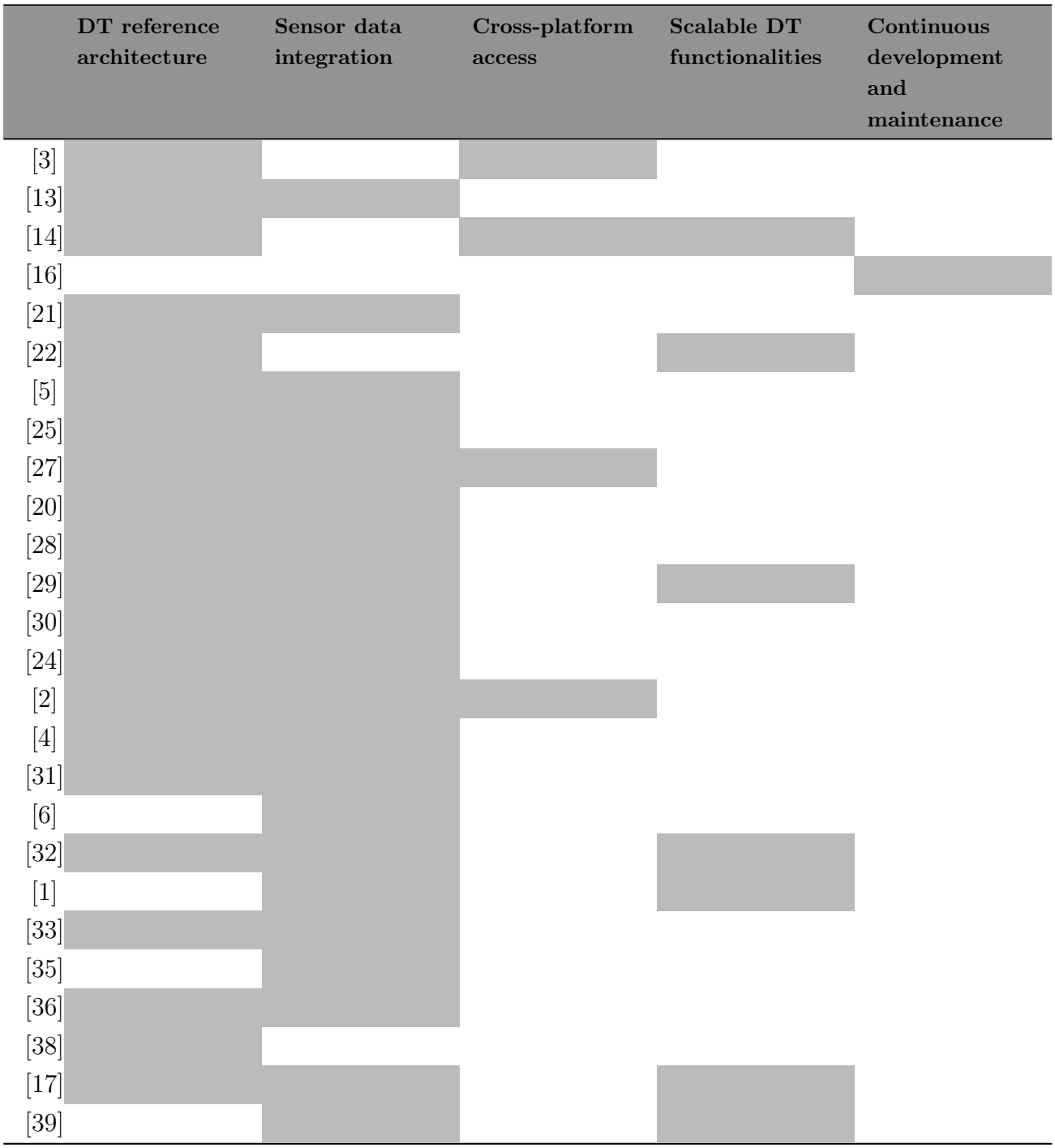
■	Within the scope of the study
	Not within the scope of the study

1.5 Summary of existing studies

DT data models, DT reference architectures, and DT applications have been investigated and studies relating to them have been analysed. These research categories were identified in the background (Section 1.1) and are shown in Figure 1.3.

The analysis done in this chapter has been summarised in Table 1.4. For the DT data models category, investigated in Section 1.2, all the studies either focused on topological data models or geometric data models with additional metadata.

Table 1.4: Summary of overall literature analysis



Almost all the studies relating to DT data models have been shown to place emphasis on DT reference architectures. All of these studies included integration with either topological or geometric data, however, less than half focused on sensor data integration into the data model. Furthermore, only two studies from the DT data model category focused on cross-platform access and scalable DT functionalities. Only one considered continuous development

and maintenance.

A possible gap was identified in Section 1.2 for a DT data modelling method that focuses on scalable, cross-platform access to DT functionalities and that also focuses on continuous development and maintenance. Next, the DT reference architecture category was investigated.

Studies focused on extending or proposing DT reference architectures were analysed in Section 1.3. These reference architectures include where and how the DT data model should be stored, how the different services within the overall DT system interact with each other, and how users interact with the DT system and its services.

All of these studies also focused on sensor data integration, however, only two studies focused on cross-platform access. Also, only one focused on scalable DT functionalities. The gap identified for the first category correlates to the gap identified in the second category, namely that there is a gap for a DT reference architecture and DT data modelling method which enables cross-platform access to scalable DT functionalities which also includes steps for continuous development and maintenance.

Finally, studies relating to DT applications were investigated in Section 1.4. Many of the studies focused on industrial applications, such as autonomous production and automated fault detection. Many of the studies also focused on simulation applications, such as optimisation models for production services and maintenance optimisation for multi-unit systems.

Almost all the studies focused on sensor data integration as it has been discussed that sensor data is critical for the calibration of simulation models. Approximately half of these studies focused on DT reference architectures and scalable DT functionalities while none focused on cross-platform access and continuous development and maintenance.

The studies analysed throughout this chapter show that DT systems provide a unique and powerful way to interact with and gain information pertaining to a physical system using a digital system. These digital systems enable high accessibility to many previously mentioned services which lead to increased system cost and time-efficiency.

However, no study in the categories of DT data models, DT reference architectures, and DT applications have focused on cross-platform, scalable access to DT functionalities while also focusing on continuous development and maintenance of the DT system. This study aims to address this gap in research.

1.6 Need for the study

DT software for complex engineering systems require powerful desktop computers or laptops to use. This excludes many potential users from access to these DT functionalities as some of them do not own or have timely access to these devices. However, most potential users have a smartphone or small mobile device which they use to communicate.

Many studies consider DT applications for desktop computers or laptops and some considered limited applications for mobile devices. However, none of them focused on enabling scalable, cross-platform access to DT functionalities in combination with continuous development and maintenance.

Hence, a need exists to enable access for many users to DT functionalities for complex engineering systems using web applications and data models that have been optimised for low-powered devices, e.g. smartphones.

1.7 Objectives

The objectives of this study are:

- to create a data modelling method to take existing DT data and create a data model which has been designed and optimised for use on devices such as smartphones, and
- to create a development methodology for web applications enabling scalable, remote access to DT functionalities while also focusing on continuous development and maintenance.

1.8 Dissertation overview

The document is structured as follows;

- Chapter 1: Research areas are identified, investigated, and studies relating to them are analysed to identify a need and determine objectives.
- Chapter 2: Two methods are proposed to achieve the objectives stated for this study.
- Chapter 3: The results of applying the proposed methods to various case studies are presented, analysed, and discussed in order to verify the proposed methods.
- Chapter 4: A conclusion to the study is presented and future research is discussed.

2 Methodology

2.1 Introduction

The background and literature analysis from the previous chapter was used to identify the need for this study; to enable access for many users to DT functionalities using web applications and data models that have been optimised for low-powered devices, e.g. smartphones.

Two objectives for this study were proposed to address this need, as shown in Figure 2.1. Firstly, to create a data modelling method for DT data models optimised for low-powered devices. Secondly, to create a development methodology for DT web applications enabling scalable, cross-platform access to DT functionalities with additional focus on continuous development and maintenance.

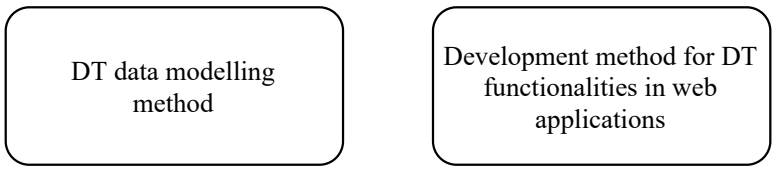


Figure 2.1: Objective overview.

These two objectives will form part of a framework to implement DT functionalities into web applications. This chapter is divided into three sections, namely a design methodology for DT data models optimised for low-powered devices such as smartphones, a development methodology for DT web applications to enable scalable, cross-platform access, and lastly, a summary for this chapter.

2.2 Design methodology for Digital Twin data models

2.2.1 Preamble

DT data models have been identified as a critical part of DT systems, as discussed in Section 1.2. A data model which is too complex may cause scalability issues, and a data model which does not contain enough information will limit the possible functionality in the DT system.

This methodology has been split into three parts, namely, the data model requirements, data model design, and data model optimisation.

The optimal data model should contain all the information which is required, without any

unnecessary data, while being modular and enabling expansion without the need for redesign. This leads to the following subsection, where the way in which the requirements are defined in this methodology are discussed.

2.2.2 Data model requirements

DT data models have been shown to consist of topological data, geometric data, and metadata [3,14,16,21–23]. The content and level of detail (LOD) necessary in each of these three categories can be obtained by defining the data model requirements.

A generic method to define the high-level scope and requirements for DT systems was proposed by Stark et al. [5] and was discussed at the end of Section 1.2.3. Their method includes a step to define the requirements for the DT data model. However, they did not go into detail regarding this.

Their method is expanded on in this subsection to create a generic method that enables the definition of the DT data model requirements, with the focus being on the required LOD within each section of topological, geometric, and metadata.

This generic method, similarly to the method by Stark et al., consists of a list of dimensions which must be defined to obtain the requirements for a DT data model to be designed. The list of dimensions to be defined is shown in Figure 2.2 with the list of dimensions by Stark et al. above, and the newly proposed dimensions specifically for DT data models below.

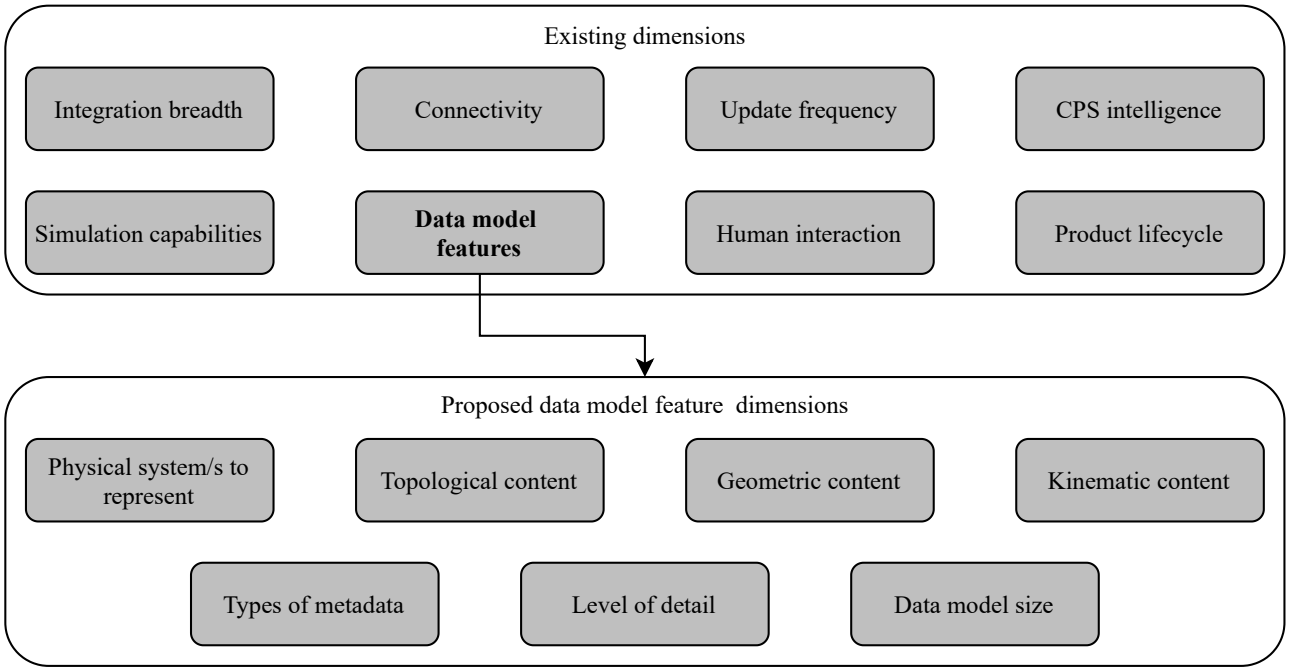


Figure 2.2: Existing dimensions and proposed dimensions.

The list of dimensions proposed by Stark et al. consists of the integration breadth, connectivity, update frequency, CPS intelligence, simulation capabilities, data model features, human interaction, and product lifecycle. Although one of their entries were related specifically to the DT data model, they did not go into detail regarding it.

The newly proposed dimensions are the physical system or systems to represent, topological content, geometric content, kinematic content, types of metadata, LOD, and lastly, data model size. The definition of the physical system or systems will be discussed first.

Physical system(s) to represent

The physical system or systems to represent is the physical entity or entities which is to be digitally represented by a DT data model. This can be any physical entity, e.g. a robotic arm, a manufacturing system layout, or a deep-level mine.

The purpose of defining this is to provide a high-level overview of the system and give context to the following entries in the proposed list. This definition leads the data model designer to start defining the scope of the data model.

The scope of the data model is the extent to which the data model represents the system. Given that the physical system to represent is a deep-level mine, the scope of the data model may vary from representing only the installed components in the mine (such as pipes and compressors) but also representing the geometry of the terrain or tunnels within the mine.

The scope of the data model must be fully defined before the data model design process is started. Drastically changing the scope during the design process may cause the data model to be redesigned as it may not be suitable for the new scope.

Topological content

Topology describes the relations between components in a system [13]. Defining the required topological content consists of defining what relations between components in a system must be stored in the data model based on the context and scope provided.

These relations can be used to describe, e.g. transport routes between bus stops in a city or the routes of pipes used to transfer water in a deep-level mine. This dimension must be defined as these relations may be required in order to achieve the purpose of the DT system.

Geometric content

The geometric content defines the 3D geometric information required for each component within the system. This should be defined based on the context and scope provided. Geometric content is required when physical shapes must be stored in the DT data model, such as the shape of a product to be manufactured.

The topological and geometric data of a system is closely linked, however, the topology of a system is a higher-level description of the system than the geometry.

Kinematic content

The kinematic content describes how components within the scene move or can move over time, e.g. the axis around and the degree to which a robotic arm can rotate or the flow of water through a pipe with a certain diameter.

Kinematic content is required when a DT data model represents a system with moving components. This kinematic content can be used to digitally visualise how the components in the system can or will move.

This is done by describing the movement possible, e.g. the degrees of freedom in a robotic arm (number of directions in which joints can move). This data can be used to simulate the performance of a system, e.g. compressed air pressure throughout a pipe system in a deep-level mine.

Types of metadata

Metadata is data about other data, e.g. time-series data collected using sensors which measure the performance of components within a system. The different types of metadata are the descriptions of all the different types which are required for the scope of the system.

Some examples of the types of metadata are: time-series data, operational data, etc. This also includes data such as documentation related to components in the system or performance data produced through simulations.

Describing the types of metadata enables the data model designer to make important considerations such as accounting for database or cloud connections with the data model.

Level of detail

The LOD is the amount of detail required in the representation of the system, e.g. detailed geometric data for product machining versus simply a box to represent the product. The LOD depends on the application as well as the scale of the system it represents. When the LOD is unnecessarily high it will lead to unnecessarily large and complex data.

The LOD within the system should be defined firstly by considering the purpose of the DT system. If the DT system will be used to simulate a manufacturing line's performance, then it is not necessary to store, e.g. the geometry of a bolt which fixes a belt feeder on the line to the floor. However, if a component is being machined, then the LOD of the geometric data of the component must be high enough to achieve the accuracy required.

The LOD also depends on the last dimension, namely, the data model size. The smaller the required data model size, the lower the LOD which can be present (given the same system to represent).

Data model size

Data model size approximations and restrictions must be defined, as this will alter the design of the data model and it also influences the feasibility of the application. Each application has different constraints (including data model size constraints) which data model designers must account for while defining this dimension.

A data model size which is not feasible to send over, e.g. mobile networks, will not be suitable for a cross-platform application as overly large data models will incur too much cost and take too much time to transfer.

The DT data model requirements are obtained by defining all of the proposed dimensions discussed in this section. The data model requirements defines what content must be present in the data model in order to enable the DT system to achieve its purpose. Defining the data model requirements is the first step in ensuring that the data model contains all the information required in the least amount of data possible.

Blume et al. [35] proposed a workflow to create DTs for technical building services operation simulation and applied their workflow to a cooling tower case study. The DT simulation system created by Blume et al. was done using a native application on a desktop computer as it required high performance computing power for the simulations.

Consider as an example that a DT system is required in order to enable cross-platform access

for users to view the near real-time performance of the cooling towers and to interact with the 3D model of the system. The existing DT data from the simulation application must be used to create the new DT data model used for cross-platform access.

Defining the data model requirements using the proposed list of dimensions in Figure 2.2 produces the following:

- physical system to represent: heat exchange system with a cooling tower array,
- topological content: routes of pipes enabling the flow of water between components,
- geometric content: shape of the cooling tower, water tanks, water pumps, and heat exchangers,
- kinematic content: expected rate of flow through pipes depending on diameter,
- types of metadata: time-series sensor data to monitor system performance,
- level of detail:
  - external shape of the cooling towers, water tanks, water pumps, and heat exchangers,
  - sensor locations (sensor geometry not required),
  - pipe routes and diameters (high detail geometry not required),
  - surrounding area geometry and landscape not required, and
- data model size: small enough not to incur high cost for mobile download or cause unfeasible download times.

The list of existing and new dimensions (specifically focusing on the data model features) should be defined by the stakeholders of the DT web application. The stakeholders are a combination of product owners, clients, and software engineers with the technical skills required to reduce the chance that unrealistic expectations are created.

After this proposed list of dimensions have been defined (as for the example above) and hence, the scope and requirements for the DT data model have been obtained, then the data model design process can begin. From the stakeholders, the software engineers will be responsible for applying the data model requirements. The proposed design method is described in the following subsection.

2.2.3 Data model design

DT data models are digital representations of physical systems and objects. Hence, an object-oriented data model design process is proposed in this subsection. The purpose of this design

method is to ensure high modularity and scalability to enable expansion of the data model features while storing only the necessary information about the system. Figure 2.3 shows the proposed method.

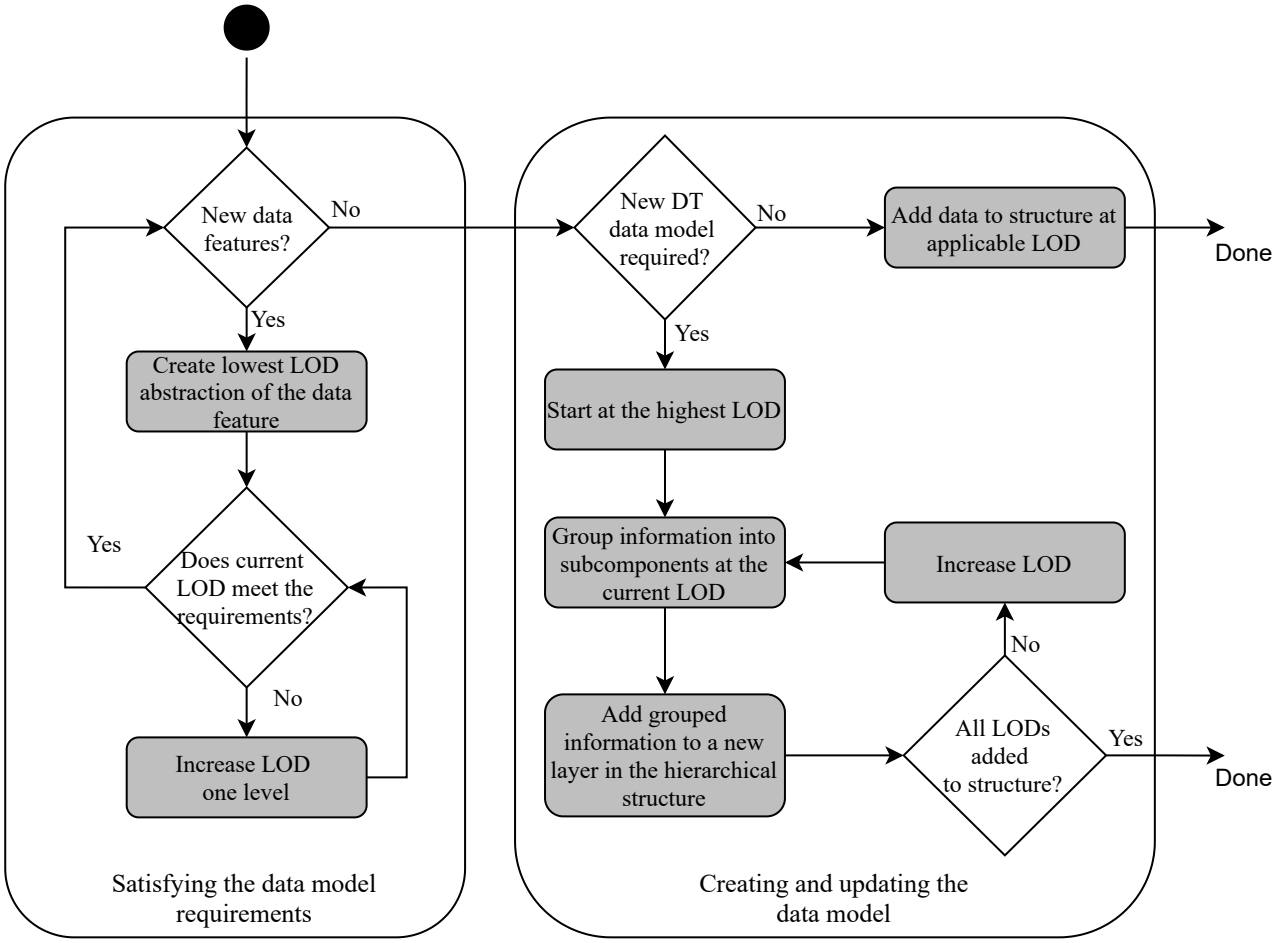


Figure 2.3: Data modelling method.

The method consists of a flow diagram of steps to complete when a new DT data model is being designed and also when a DT data model is being extended for new requirements. This method should be used after the data model requirements have been defined. This method receives DT data as input and creates a highly modular and scalable data model with only the required information present based on the requirements defined.

Satisfying the data model requirements

One of the first steps in the method to ensure that DT data models contain all the necessary information in the least amount of data is to choose the necessary information based on the requirements defined. In this way, unnecessary data in the DT data model is minimised from the start of the design process.

The first half of the method in Figure 2.3 is to iterate over all the new features present in the received DT data and to determine their relevance and LOD required to satisfy the

requirements. The more the LOD is increased, the more data is required to store the information.

Obtaining the information required from the data is done by starting with a new or updated feature in the received data, at the lowest LOD possible, and identifying what information is present. The requirements applicable to the current subcomponent is then tested against the current information.

If the requirements are not satisfied, then the LOD is increased, and the information present at the new increased LOD is identified. The requirements are then tested again. This iteration through increasing LODs continues until the requirements are satisfied.

Once this is the case, then no additional LOD data is added for the data feature. Abstraction is the process of simplifying detailed information into a high-level overview of the data [23], e.g. instead of storing the exact geometric shape of a pipe, only the route that the pipe takes and its diameter are stored. Abstraction greatly reduces the required data model size used to store the necessary information based on the requirements.

When the requirements are not satisfied, the LOD is increased, and the data parameters are identified and linked to requirements. This continues until the requirements have been satisfied. When no new data features are present, the new DT data model is either created or updated.

Referring back to the example of the cross-platform DT system for the cooling tower exchange system: with the data model requirements defined, as shown in the previous section, the process to satisfy the data model requirements as proposed in Figure 2.3 and discussed above can be completed.

As this will be the first iteration, all the data features will be new and the first feature chosen is the cooling tower geometry. According to the proposed method, the data must be abstracted to the lowest LOD and analysed. Abstracting the geometry to the lowest LOD would be to represent the cooling tower geometry as a simple bounding box of the actual geometry.

The next step is to test this LOD against the data model requirements. The external geometry of the cooling tower is required and hence, this first LOD (a bounding box) does not satisfy the requirements. The LOD is therefore increased to include the external geometry of the cooling tower and all other further increased LOD data is abstracted.

The requirements are then tested again and as the requirements are now satisfied, no additional LOD data is added for the data feature. The next data feature example is the

pipes within the system. The lowest level of detail for the pipes are simply the route which they take in the system.

The requirements are tested and it is seen that the diameter of the pipes are also required. The LOD is therefore increased to also include the diameter of the pipes and the data model requirements for this feature have been satisfied.

This process continues until all the new data features have been tested against the data model requirements. This process ensures that the least amount of data is used to satisfy the data model requirements. Once the requirements have been satisfied, the next part of the method is creating or updating the model.

Creating and updating the data model

During creation, the DT data model will be structured using the hierarchical object-oriented JavaScript Object Notation (JSON) format, as this format follows the object-oriented paradigm and enables integration between many different types of systems.

This format is also highly modular which enables the addition or alteration of data model features when necessary without the need to redesign the model. The modularity is one of the most important features of this format.

The second half of Figure 2.3 shows the proposed method to create and update the DT data model. In the first iteration, when the data model is created, the most comprehensive LOD in the data (identified using the requirements) is chosen and grouped into subcomponents. These initial subcomponents are then used to create the initial JSON structure.

If more LODs are present in the identified data, then the LOD is increased and the information at the current LOD is grouped into subcomponents and added to the hierarchical structure. This process is repeated until all of the required data has been added to the structure.

Using the cooling tower DT system example created from the study done by Blume et al. [35], the proposed data model creation method is illustrated in Figure 2.4. Figures 2.4a and 2.4b show the result of the first and second iteration of the data model creation method.

The first iteration of the data model creation produces a high level overview of all the relevant components within the system, as shown in Figure 2.4a. The resulting data model contains the minimum required LOD about each component as all other increased LOD data has been abstracted for the current iteration.

As defined by the requirements, increased LOD for certain components are required, such as the cooling towers. The external geometry of the cooling towers is required, and therefore, another iteration of the method is required for which the result is shown in Figure 2.4b. The external geometry and centre point of each cooling tower is now present in the data model.

```
{
    "Cooling Tower DT Monitoring System": [
        {
            "Name": "Cooling towers",
            "Parameters": "Sprays water
            through air which is blown
            upwards  with fans to transfer
            heat from the water to the
            atmosphere"
        },
        {
            "Name": "Pipes",
            "Parameters": "Enables the
            transfer water between heat
            exchangers, water tanks, and
            cooling towers"
        },
        {
            "Name": "Heat exchangers",
            "Parameters": "Transfers heat from
            system components into water in
            pipes"
        },
        {
            "Name": "Water tanks",
            "Parameters": "Stores water"
        },
        {
            "Name": "Water pumps",
            "Parameters": "Pumps water through
            pipes"
        }
    ]
}
```

(a) Result from the first iteration.

```
{
    "Cooling Tower DT Monitoring System": [
        {
            "Name": "Cooling towers",
            "Entities": [
                {
                    "Name": "Cooling tower A",
                    "External geometry": [
                        {
                            "x": "80m",
                            "y": "80m",
                            "z": "150m"
                        },
                        {
                            "x": "78.5m",
                            "y": "80m",
                            "z": "150m"
                        }, ...
                    ],
                    "Centre": {
                        "x": "300m",
                        "y": "50m",
                        "z": "75m"
                    }
                },
                {
                    "Name": "Cooling tower B",
                    "External geometry": ...
                }, ...
            ]
        },
        {
            "Name": "Pipes",
            "Entities": [
                {
                    "Diameter": "230mm",
                    "From": "Heat exchanger 1",
                    "To": "Cooling tower A"
                },
                {
                    "Diameter": "220mm",
                    "From": "Cooling tower A",
                    "To": "Water tank A"
                }, ...
            ]
        }, ...
    ]
}
```

(b) Result from the second iteration.

Figure 2.4: Data model creation example for two iterations.

The data model in Figure 2.4b also shows slightly increased LOD data for the pipes in the system. The external geometry data of the pipes are not stored, as for the cooling towers, only the diameter and from- and to-coordinate data of the pipes are required. The rest of the resulting data model has been omitted for brevity.

Once the DT data model has been created and another data feature is to be added, then the DT data model only needs to be updated as it is modular. This process is defined as taking the new data model feature which has been analysed according to the updated requirements, choosing the appropriate subcomponent and LOD within the DT data model, and inserting the data at that point.

The proposed methods to define the DT data model requirements and to create and update the DT data model have been discussed. The final step in the design methodology for DT data models is data model optimisation for use on mobile devices such as smartphones. This is discussed in the following subsection.

2.2.4 Data model optimisation

Devices such as smartphones do not have the high levels of computing power of devices such as desktop computers which are typically used for DT systems, as shown in the previous chapter. This means that the DT data model must be optimised in order to avoid unnecessary complexity and required size based on the data model requirements set by the proposed method earlier in this section.

Optimisation is the process of making the most effective use of a resource [41]. This optimisation process should, however, not reduce the scalability nor modularity of the DT data model. Figure 2.5 shows the proposed optimisation process which is the last part of the overall design methodology for DT data models.

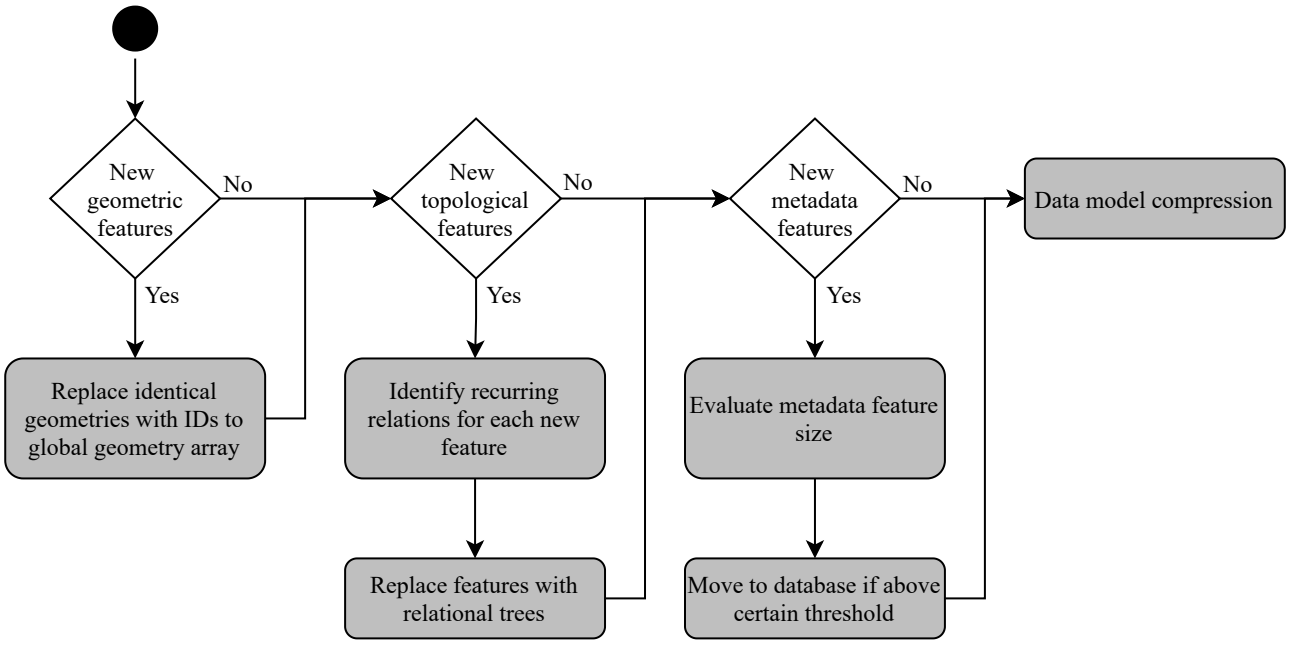


Figure 2.5: Data modelling optimisation.

The proposed method iterates through the three types of data features present in the DT data model, namely geometric, topological, and metadata features. Firstly, the geometric data features within the data model are investigated to identify any identical geometries.

As the LOD for each component will vary due to the data modelling process proposed in the previous section, it may be the case that many different components in the system have identical geometries. Referring back to the cooling tower example, if all the heat exchangers have the same geometry, then the same geometry data will be stored in multiple components within the data model.

Storing the same geometry in many different places in the data model causes the data model to be larger than necessary. This issue, if not resolved, may cause unfeasible data model sizes for complex systems such as complete underground mines with thousands of components.

The identical geometries should rather be removed from the applicable components and added to an array of geometries shared throughout the DT data model. Each applicable component then only receives a link, e.g. an ID, with which it can be used as reference to the geometry.

As identical geometries are highly likely present in the DT data model, especially for large systems, the data model size difference from applying this first step is likely to be significant. The topological features are considered next.

Topological data, as defined in the previous chapter, describes relations between components in a system. These relations may be the distance between two components or it may be the route that a set of pipes take from one component in the system to another.

Recurring relations simply mean that many objects in a system have similar relations to other objects in the system. An example of this is all the pipes present in a deep-level underground mine.

Consider that each section of pipe in the system are modelled as separate entities, with a diameter and a from- and to-coordinate. This would mean that for each pair of connected pipes, one coordinate is stored unnecessarily as both of the pipes contain the same coordinate.

The proposed method to avoid this is to model all the separate pipe entities as one tree structure. A tree consists of edges and vertexes, which for this example, are the pipes and the coordinates where the pipes connect respectively.

Some of the advantages that the tree structure provide are that each coordinate is only stored once in the data model and features such as tree searching algorithms can be used on the data. Tree searching and traversal algorithms are widely known for their efficiency which is a

highly desirable advantage when computing power is a limiting factor.

Finally, the metadata features are considered. Metadata can consist of many different data types, such as time-series sensor data and Portable Document Format (PDF) documents relating to a component's maintenance schedule or installation detail.

The next step of this method is to evaluate whether the metadata should be moved to a database in order to reduce DT data model size, as this metadata may increase the data model size unnecessarily and the data may not be required as frequently as, e.g. the geometry of a component.

If the metadata is deemed too large or is not required frequently, then the metadata should be stored in a database with a link stored in the DT data model. This link can then be used, once the metadata is requested, to query the metadata from a database.

After each one of the three types of data have been iterated over and the proposed process has been completed, the final data structure is obtained. The last step of this method is data model compression.

There are many different compression standards and it depends on the specific application for which this method is used, however, a widely used compression standard is the Zip format. As the DT data model will contain much uncompressed text, the Zip compression will yield a high compression ratio.

The proposed data model optimisation which has been discussed was the last of the three parts of the design methodology for DT data models. A summary of the overall design methodology is given in the following subsection.

2.2.5 Verification

Verification is required to ensure that the correct system is being digitally represented by the DT web application. This is a high-level analysis of the correctness of the digital representation of the physical system.

As seen in Figures 2.3 and 2.5, the only source of data for the method is that of the original input data model. Hence, there is no opportunity for external data to change the information contained in the original data model.

The resulting DT data model from using the methodology may change the LOD of some information within the data, the data structure will also be different, and the format in which

the information is stored may be different. However, no data will be changed by the use of external data sources.

This closed loop verifies that the correct data is present in the resulting DT data model when using the proposed methodology. The methodology will be summarised in the following subsection.

2.2.6 Summary of the design methodology for Digital Twin data models

The methodology in this section was proposed to address the first objective of this study. This first objective was to create a data modelling method to take existing DT data and create a DT data model which has been designed and optimised for use on devices such as smartphones.

The proposed methodology, summarised in Figure 2.6 within the overall framework, was split into three parts, namely, the data model requirements, the data model design, and the data model optimisation. The data model requirements definition consisted of defining a list of dimensions, shown in Figure 2.2, in order to obtain the requirements for a DT data model.

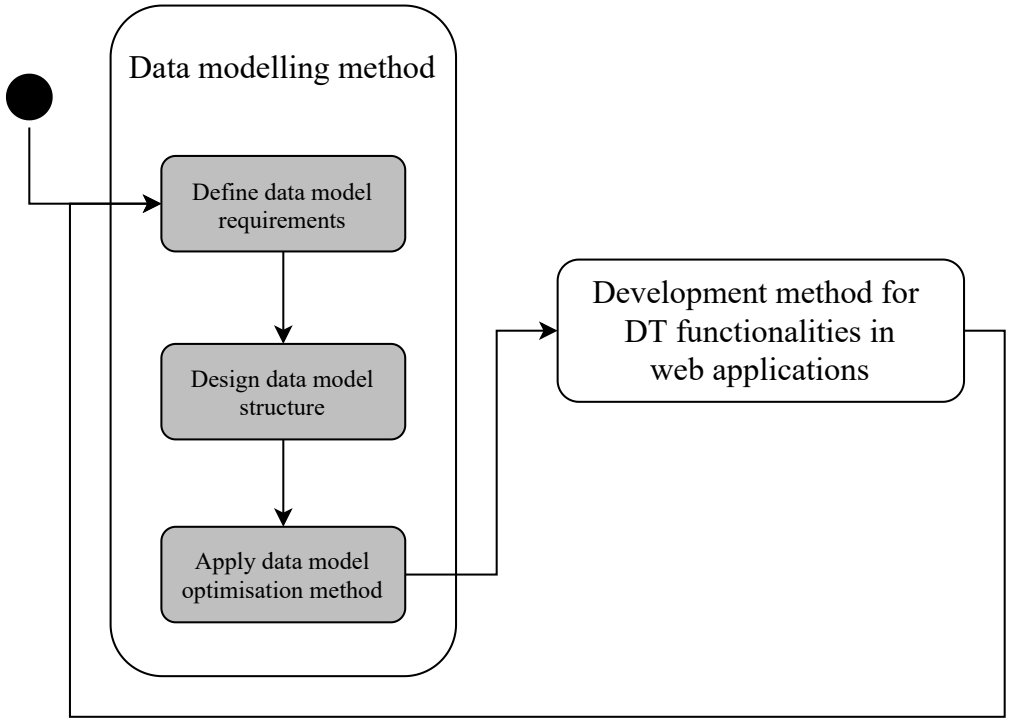


Figure 2.6: Summary of the proposed method within the overall framework.

Defining the proposed list of dimensions not only produces the requirements of the DT data model, but also provides the initial scope for the DT system. After the requirements have been obtained, the second part of the method, the data model design, can be applied.

The data model design method, shown in Figure 2.3, consisted of taking the existing DT data, abstracting and removing data based on the requirements defined, and finally, creating or updating the new DT data model. This method was proposed to create a data model which contains all the information needed to satisfy the requirements while producing a scalable and modular structure which can be updated at a later stage without the need for redesign.

The final part of this methodology is the data model optimisation, shown in Figure 2.5. It consisted of iterating over each type of data present in the data model and ensuring that the data model firstly, does not contain identical data in multiple places (such as identical geometries), and secondly, enabling efficient searching and traversal of relational data.

The final step in the optimisation process is data model compression which further reduces the overall size of the data model. Reducing the complexity and size of the data model is an important improvement as large data transfer over mobile networks is costly in both time and money.

The proposed design method for DT data models has now been discussed and the next step in the overall framework is the development method for DT web applications.

2.3 Development method for Digital Twin web applications

2.3.1 Preamble

Web applications enable cross-platform access for many users to system functionalities. This means that any user that has a device that supports a web browser can access these functionalities. Web applications are, however, not as performant as native applications [11].

For this reason, the development method for computationally intensive web applications enabling access to DT functionalities, proposed in this section, will focus on scalable and modular implementation as well as continuous development and maintenance.

This section consists of five subsections, namely, the scope of functionalities, web development library selection, design of the required functionalities, scalable and modular implementation, and lastly, continuous development and maintenance.

In the first subsection, the process to define the scope of functionalities for DT web applications

will be discussed.

2.3.2 Scope of functionalities

The scope of a system must be defined before it is designed. If this is not done, systems will often need to be redesigned as it may not be possible to achieve the new scope of the system as it was originally designed.

System redesign due to scope creep costs time and money. This can be avoided by defining in detail the scope of a system before it is designed. The method to define the DT data model requirements and obtain the initial scope for DT systems were proposed in Section 2.2.

The method proposed by Stark et al. [5] was extended in Subsection 2.2.2 to include a more detailed definition of the data model requirements. Their method, and more specifically, their list of dimensions will be applied here to define the scope for web applications enabling access to DT functionalities. Stark et al. applied their method to a native application case study.

Table 2.1, which follows, describes each of the dimensions and provides examples for each. The dimensions listed in Table 2.1 have been discussed at the end of Section 1.2. All of these dimensions together are aimed at defining the scope of a DT system.

Each of the dimensions described in Table 2.1 must be defined by considering the aim and purpose of the DT system as well as the DT data model requirements defined using the method proposed in Section 2.2.2. The scope of DT systems can be obtained by defining this list.

Table 2.1: Dimensions proposed by Stark et al. [5]

Dimension	Definition	Example
Integration breadth	How much context of the physical system is desired	Full environment, system components, single component, etc.
Connectivity	Flow of data between the physical system and DT system	Unidirectional or bidirectional
Update frequency	Frequency at which data must be synced between the physical and digital system	Near real time, hourly, daily, etc.
CPS intelligence	Ability of the DT to control or enable control of the physical system	Autonomous or human triggered
Simulation capabilities	What simulation features do the DT system provide	Dynamic, steady-state, or none

Table 2.1: Continued from previous page

Dimension	Definition	Example
Data model features	Topological, geometric, and metadata contained in the data model	External geometry of component, component position, etc.
Human interaction	How will users interact with the DT system	VR/AR, smart devices, web applications, etc.
Product lifecycle	For what stage in the product's lifecycle will the DT system be designed	Production, assembly, operational stage, etc.

The next process in the methodology, namely web development library selection, is discussed in the following subsection.

2.3.3 Web development library selection

Front-end web development libraries are resources and tools used by software developers to build and maintain web applications. These web development libraries promote code re-use, e.g. the use of standardised libraries for database access and session management.

These libraries may provide templates, boilerplate code, classes, and optimised functions to promote code re-use and enable the efficient authoring of high quality systems [42]. Web developers enhance the performance of their web application more often than not by using these optimised built-ins and functions, as reputable libraries are often implemented by highly-skilled and experienced developers [43]. These optimised built-ins and functions are also peer reviewed on source control websites such as GitHub.

Many web development libraries have been created and some have also been deprecated due to poor implementation or design flaws. Many libraries have, however, become widely used and are expanded on further and maintained. At the time of this study one of the most popular libraries is React. Its initial release was in 2013 and at the time of this study, almost eight years later, it has shown significant growth in features and popularity[1].

These libraries provide higher-level development using optimised functions. The average development time for applications drastically decreases when these libraries are used versus when they are not used.

These libraries, having already implemented much of the functionality that a web application

[1]https://blog.jetbrains.com/dotnet/2020/06/16/developer-ecosystem-2020-key-trends-c/

requires, also drastically reduce the amount of new code that a developer needs to write to achieve a requirement. Further, multiple different libraries can be integrated into one web application.

Each application does, however, have different requirements which affects the choice of web libraries to use. Specifically for DT systems, one of the most important requirements are tools for rendering 3D objects in a web browser, e.g. the Web Graphics Library (WebGL).

Web development libraries must be chosen based on the scope of the DT system, the performance of the library, and the designed data model.

2.3.4 Design of required functionalities

The scope and required functionalities have been defined in the previous subsections. The process to design these required functionalities is discussed in this subsection. Their design greatly affects the scalability and modularity of the DT system they will be used in, as shown in Section 1.2.

Figure 2.7 shows the proposed process to design the required functionalities for DT web applications. This process forms part of the greater development method for DT web applications. The process starts after the DT data model has been designed, the scope of the web application has been defined, and the web development libraries have been chosen.

The first part of the process is the object class selection and the second part is the design of the required functionalities based on the selected or created classes. Classes have been chosen as data structures as they are one of the core concepts of object oriented programming.

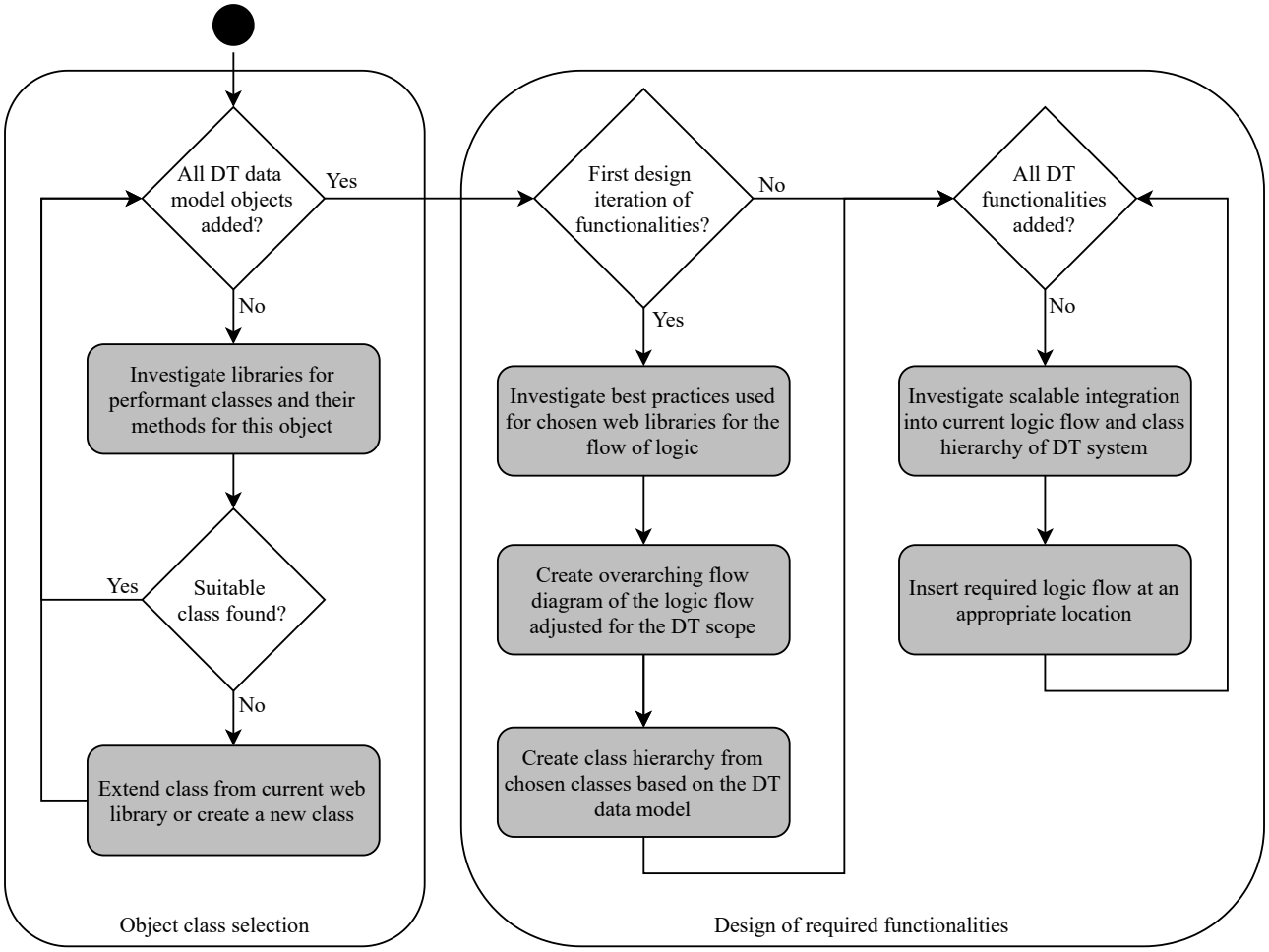


Figure 2.7: Designing the required functionalities.

Object class selection

All the different types of DT data model objects present in the DT data model, which have been created using the proposed method in Section 2.2, are iterated through and appropriate classes are selected or created for them. Different types of objects are, e.g. a cooling tower object type or a pipe object type.

This is done by firstly investigating the current web development library for an appropriate class which fits the attributes of the object in the data model. Then the class and its methods' performance are analysed to determine whether or not the class will be adequately scalable.

If a suitable class is found, then a reference to the class is stored for the current object and the next type of object in the data model is considered. A fully suitable class is one that contains all the attributes required by the DT data model object and also contains performant methods with which the objects can be manipulated or searched for.

If no classes for an object can be found in the web development library, then a suitable base

class from the library should be extended or a new class must be created. Suitable base classes are classes that contain some attributes of the current DT data model object, but not all of them. These base classes can be extended to also include the other attributes in the object.

New classes must be created if suitable base classes cannot be found. New classes may also require new methods which will be designed in the next part of the process, namely, the design of required functionalities.

Design of required functionalities

All the required functionalities are iterated through, the logic flow diagram is created, and the class hierarchy is defined in this part of the process. The overarching flow of logic must be defined and the initial class hierarchy must be created in the first iteration of the overall methodology.

The best practices for the web development library should be investigated to find a suitable high level logic flow. A suitable logic flow is one that enables modular development and scalable implementation.

Best practices should be used as many other developers have used and contributed to the web development library and therefore may have a more in-depth understanding and experience with the library. This high level logic flow must also be suitable for the DT scope.

The class hierarchy should then be established. This is done by using the already defined object hierarchy in the defined DT data model.

The logic flow is then extended to account for all the required DT functionalities. This is done by investigating the scalable integration into the current logic flow and class hierarchy.

The functionality may require the frequent searching for objects in the system, for which an initial sorting and indexing approach may be optimal. This means that future searches can use algorithms such as binary tree search methods which greatly increase search speed and overall system performance.

Lastly, the required logic flow is inserted into the appropriate location based on the result from the integration investigation already completed. This is the process of defining which steps need to be taken to achieve the required functionality.

The process of the design of the required functionalities has been discussed and the next part of the overall DT web application design method is the scalable implementation of the designed

functionalities.

2.3.5 Scalable and modular implementation

The third part of the development method for DT web applications is the scalable and modular implementation of the functionalities designed in the second part of this method. This will be based on the scope of the DT system established in the first part of this method.

The second objective of this study is to create a development method for DT web applications to provide access for many users to DT functionalities. If the performance of the system deteriorates drastically as the number of users increases and the physical objects the system represents grow larger, then the web application will not be scalable and the objective will not have been met. A scalable implementation of the required functionalities ensures that this will not occur.

It should be expected that some performance will be lost on extremely large systems and extreme numbers of users, however, the scope of the system defines what is expected from the DT web application. Lower LOD data models must be designed if the size of the digital representation of the physical system creates an unfeasible expectation of performance from the web application.

Figure 2.8 shows the scalable and modular implementation process proposed. The process consists of firstly grouping the logical flow into modular functions. These functions, if removed, should not affect the working of the other remaining function. This produces modular functions.

After the grouping has taken place, then for each of the functions, an investigation is done to identify existing functions in the chosen web library. Existing functionality in the web library enables code re-use as well as the option to use well optimised code that, in most cases, has been peer reviewed.

If no appropriate functions are found, then applicable industry standards should be investigated in an attempt to find a well-tested and peer-reviewed approach. Industry standards are implementation methods or approaches which have been identified, tested, and proven by industry specialists to be the best approach to solve a problem.

There are many cases where no industry standard has been created. However, problems experienced by many people are likely to have been discussed and documented. These discussions and documentation are valuable resources to build upon already existing knowledge and to avoid spending time on the re-implementation of functionality. These discussions and documentation lead to the creation of industry standards.

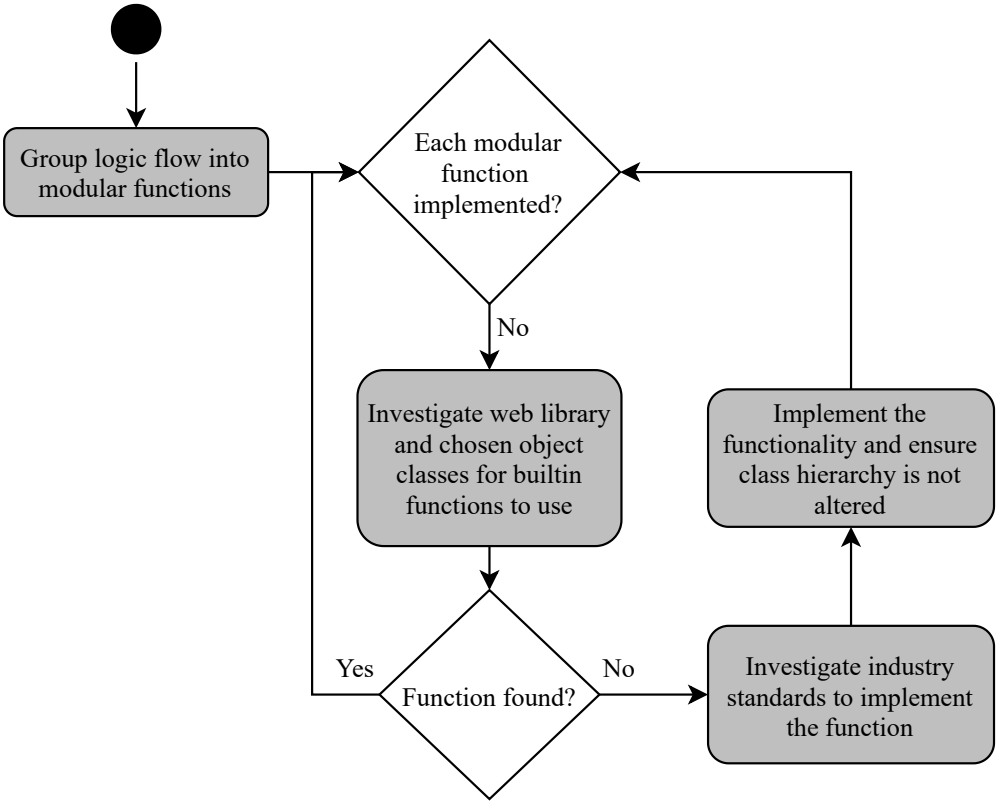


Figure 2.8: Implementation of scalable and modular functionality.

Finally, after the industry standards have been investigated, the functionality can be implemented. This must be done in such a way that the class hierarchy is not altered and if the function were to be removed, then the other functionality should still work. This produces modular functionality.

The scalability is achieved through the use of existing built-in library functions and the investigation of and, if appropriate, utilisation of industry standards to implement the functionality.

2.3.6 Continuous development and maintenance

Stagnated development and maintenance on software systems are two of the main factors causing software systems to be classified as legacy systems. These are systems built using outdated technologies and methods which require more time to maintain and to add new features [44].

Over time, less and less users will use these legacy systems as newer and more optimised systems are created. Eventually, if the functionality that the system provides is still required, then many hours of development time and hence, money, must be spent to re-implement these systems.

This can, however, be avoided if the software systems are regularly maintained and additional functionality is added. Regular maintenance is aimed at checking whether the functionalities of a system have been implemented using the latest industry standard and whether new research shows even further improvement beyond the industry standards.

The time spent on this continuous maintenance should not cause further development to halt. Performance analysis should be used to identify the parts of the system which show the greatest potential for improvement.

The proposed process in Figure 2.9 is based on the Plan, Do, Check, Act (PDCA) cycle which forms part of the ISO 9001 standard [45]. The loop which creates the cycle is not shown in this section, however, the cycle is shown and described as part of the overall framework in the following section.

The process is divided into two branches, namely, a branch to improve performance of the system and a branch to resolve an issue within the current functionality of the system. The first branch to be discussed is the performance improvement branch.

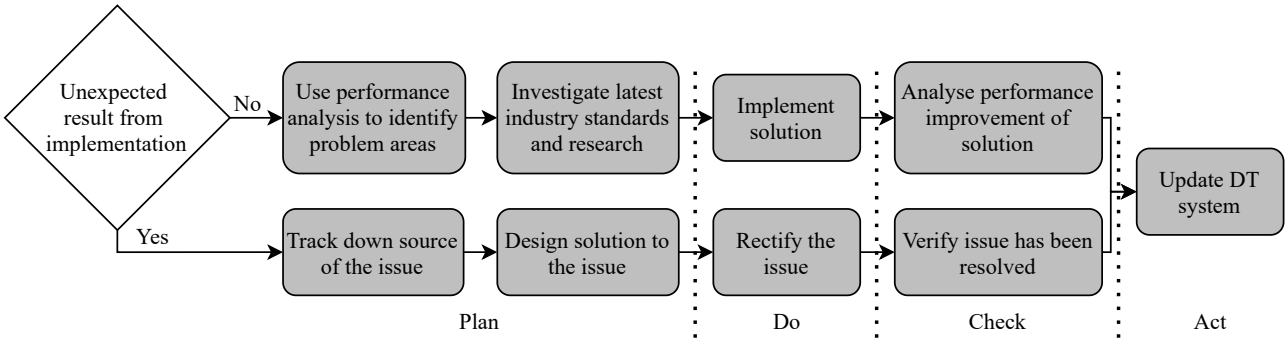


Figure 2.9: Continuous development and maintenance process.

Enhancing performance

As discussed, continuous development in order to enhance the performance of a software system ensures that it does not become a legacy system. It also enables the system to be built using up-to-date methods and optimised industry standards. It also enables the possibility of including the latest methods in research into the functionality of the system to further improve performance.

The top branch of Figure 2.9 shows the process. The first section of the branch is the planning phase where performance analysis of the current system is done. An investigation is also done in the planning phase in order to see if there are new industry standards or research for the areas which have been identified with high potential for improvement.

If an improved industry standard or new research has been found, then the current performance of the system is compared to the expected performance of the new method found. If the increase in performance is significant enough to justify the expected implementation time, then in the next step of the process, the new method is implemented.

After the implementation, during the check phase, the actual performance of the new method is measured and analysed. If the performance of the new method is greater than the old method, then during the act phase, the DT system is updated with the new method.

These steps form part of the greater framework for development method for DT web applications. The loop which connects the final act phase to the initial plan phase is shown in the summary of the framework. The bottom branch of the process is discussed next.

Resolving issues in current functionality

It is often the case that issues are found in software systems. These may consist of small interaction issues that a user may experience, or a calculation error. Many systems do use automated testing to minimise the likelihood that this occurs, however, these issues still arise despite all the possible tests.

The bottom branch in Figure 2.9 shows the proposed process to solve these issues. The first phase, as with the other branch of the process, is the planning phase. During the planning phase, the source of the issue is searched for, identified, and the solution to the issue is designed.

In the next phase, the issue is then resolved by implementing the solution. Following this, during the check phase, the system is tested in order to determine if the solution was successful. If found that the issue has been resolved, then the DT system can be updated with the solution.

This process, as stated, forms part of the overall framework and hence, the loop to close the cycle will be shown in the summary of the framework. The above-mentioned section was the last part of the continuous development and maintenance process and also the larger development method for DT web applications. In the following section, the method will be summarised.

2.3.7 Summary of the development method for Digital Twin web applications

The methodology in this section was proposed to address the second objective of the study. This second objective was to create a development methodology for web applications enabling scalable, remote access to DT functionalities while also focusing on continuous development

and maintenance.

The summary of the proposed method within the overall framework is shown in Figure 2.10. The method was split into three parts, namely, designing a solution, implementing the functionality, and identifying lacking performance or functionality to improve and optimising the identified functionality.

The first part, designing the solution, was further broken down into the processes of defining the scope of the required DT functionalities, the selection of appropriate web development libraries, and the design of the required functionalities. The process of defining the scope consisted of using the dimensions proposed by Stark et al. [5] as well as the DT data model requirements which can be defined using the dimensions proposed in Subsection 2.2.2.

The process of web development library selection was proposed to enable the re-use of peer-reviewed and tested implementations of functionalities. This process also aims to prevent the re-implementation of existing functionality, which wastes time and money.

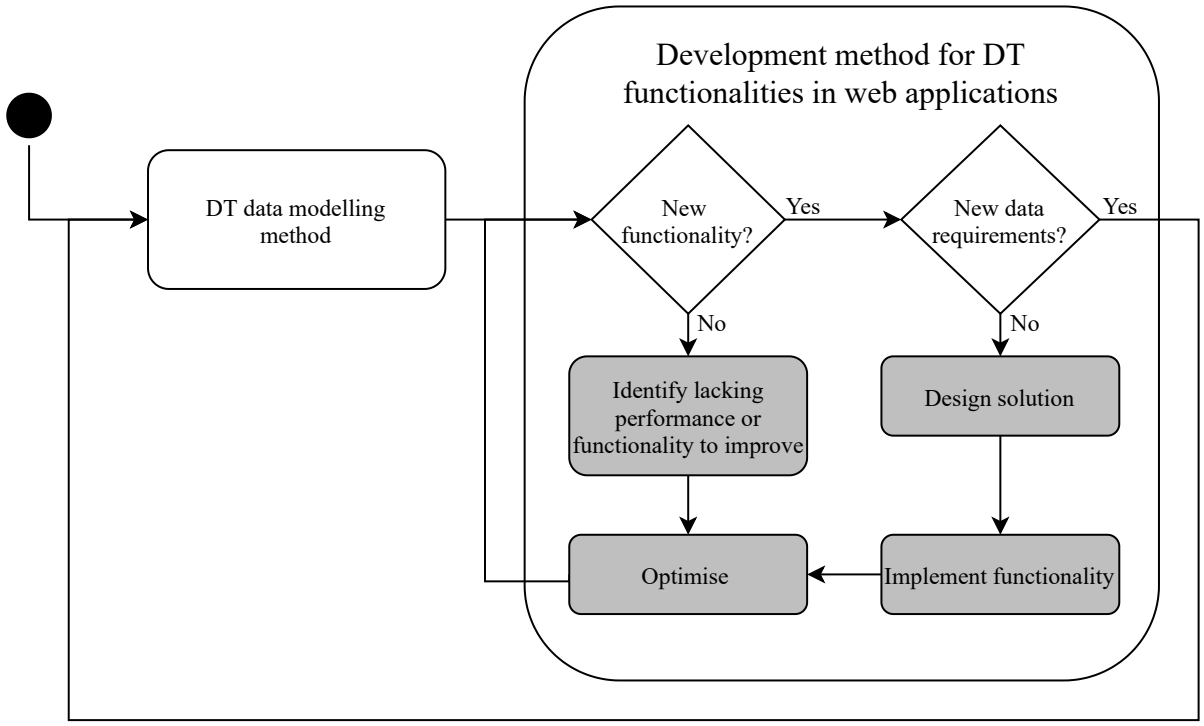


Figure 2.10: Summary of the proposed method within overall framework.

The final step of designing the solution consisted of designing the functionalities. It consisted of assigning classes to each hierarchy in the DT data model which can be designed using the method proposed in Section 2.2.

Designing the functionalities also consisted of investigating industry standards and the latest research to create the logic flow of the required functionalities. The proposed method to

designing the solution focuses specifically on the scalable and modular design of the DT functionalities.

The second part of the development method is the implementation of the required functionality. This consisted of grouping the logic flow into modular groups, investigating the web development library for built-in functions, investigating the latest research for optimised methods, and lastly, the scalable and modular implementation of the functionality.

The process to implement the functionality was aimed at retaining the class hierarchy created and creating modular and scalable functions to achieve the required functionalities. The third and final part of the method was the continuous development and maintenance of the DT system.

The process of continuous development and maintenance was proposed to ensure that the DT system remains up to date with the latest industry standards as well as the latest research and best practices. The inner loop in Figure 2.10 shows the PDCA cycle on which the continuous development and maintenance process was based.

The process is specifically aimed at reducing the likelihood of the system becoming a legacy system and, more importantly, ensuring that the system uses state-of-the-art methods for improved performance. This process was also proposed to create an opportunity for new research to be conducted and enable contribution to engineering knowledge.

Now, with the second method discussed which aims to address the last objective of this study, the overall framework for developing DT web applications will be discussed. It consists of utilising the two methods already discussed in this chapter.

2.4 Framework for developing Digital Twin web applications

The two methods proposed in this chapter together form the framework to implement DT functionalities into web applications. This is shown in Figure 2.11. The data modelling method is shown on the left. The development method for DT functionalities in web applications is shown on the right.

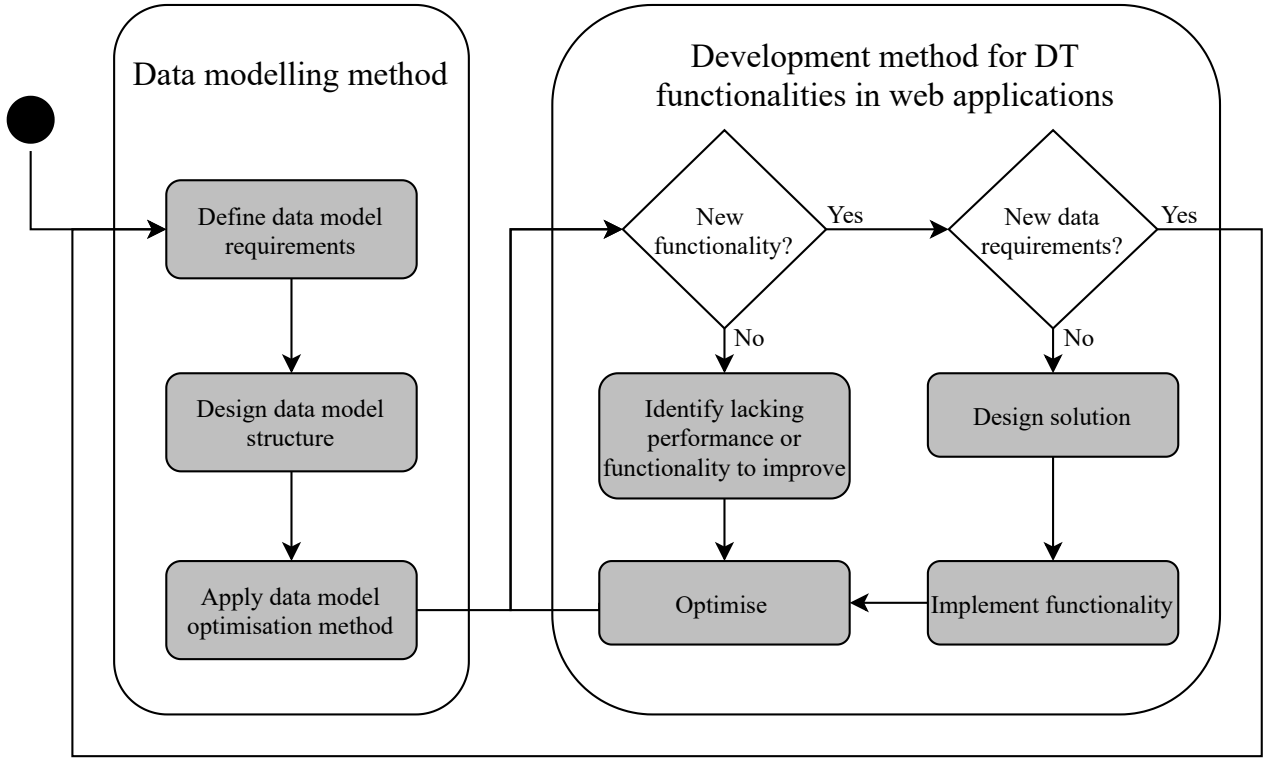


Figure 2.11: Framework to implement DT functionalities into web applications.

The data modelling method was proposed to address the first objective of this study. This first objective was to create a data modelling method to take existing DT data and create a DT data model which has been designed and optimised for use on devices such as smartphones.

The development method for DT functionalities was proposed to address the second objective of the study. This second objective was to create a development methodology for web applications enabling scalable, remote access to DT functionalities, while also focusing on continuous development and maintenance.

The framework starts by designing a modular data structure which contains only the information necessary based on the defined requirements. Following this, as this is the first iteration, the design and implementation of all the required functionalities is done.

This is followed by an optimisation step which is aimed at increasing the performance of problem areas within the system. This starts the PDCA cycle, where if no new functionality is required, then lacking performance or functionality can be identified within the system and the relevant part of the system can be optimised.

This PDCA cycle continues until new functionality is required. If new data model content is required, then the data modelling method is used to add or change the required information at the appropriate hierarchy level within the existing DT data model.

If no new data requirements are necessary, or if the data modelling method has been used to add or change the relevant information, then the new functionalities can be designed and implemented using the second part of the framework.

The framework was proposed to enable scalable access for many users to DT functionalities using web applications. DT systems have been shown in the previous chapter to require powerful devices such as desktop computers in order to enable access for users to the functionality.

The framework is aimed at reducing this high computing power requirement such that mobile devices, e.g. smartphones, can be used to access the DT functionalities. Therefore, enabling access for many more users to the functionalities on cross-platform applications.

In the following section, the summary for the proposed methods and framework are discussed. These processes will be used to evaluate the results of the case studies and validation done in Chapter 3.

2.5 Conclusion

The background and literature analysis from the previous chapter was used to identify the need for this study. This is to enable access for many users to DT functionalities using web applications and data models that have been optimised for low-powered devices, e.g. smartphones.

Two objectives for this study were proposed to address this need, as shown in Figure 2.1. Firstly, to create a data modelling method for DT data models optimised for low-powered devices. Secondly, to create a development methodology for DT web applications enabling scalable, cross-platform access to DT functionalities with additional focus on continuous development and maintenance.

The data modelling method for DT data models, summarised in Figure 2.6 within the overall framework, was split into three parts, namely, the data model requirements, the data model design, and the data model optimisation.

The requirements enable the definition of what information was necessary for the DT data model. The data model design process enables the creation and updating of the DT data model in a modular and scalable manner.

The data model optimisation is aimed at reducing the complexity of the data model and reducing the data model size. All of these processes together were proposed to enable the

creation of a DT data model that contains all the necessary information in as small data model as possible.

The summary of the development methodology for DT web applications within the overall framework is shown in Figure 2.10. The method was split into three parts, namely, designing a solution, implementing the functionality, and identifying lacking performance or functionality to improve and optimising the identified functionality.

The first part, designing the solution, was proposed to prevent the re-implementation of existing functionality, which wastes time and money. This process is also aimed at designing new functionality based on the scope of the system, which is also defined using this process.

The process to implement the functionality was proposed to enable the use of industry standards and the latest research for the functionality required. A modular and scalable implementation is also one of the main focuses of this process.

The continuous development and maintenance was proposed to ensure that the DT system remains up-to-date with the latest industry standards as well as the latest research and best practices. The inner loop in Figure 2.10 shows the PDCA cycle which the continuous development and maintenance process was based on.

The process is specifically aimed at reducing the likelihood of the system becoming a legacy system and, more importantly, ensuring that the system uses state-of-the-art methods for improved performance. This process was also proposed to create an opportunity for new research to be conducted and enable contribution to engineering knowledge.

The two methods proposed were verified using various case studies and experiments. The results from the verification show that the methods both correctly achieve the objectives for which they were proposed.

The verification process showed promising initial results from the two methodologies of the framework. The framework was applied to various case studies of different deep-level mines and the results will be presented, analysed, and discussed in the next chapter.

3 Results

3.1 Preamble

The results presented, analysed, and discussed in this chapter were obtained by using the proposed framework to create a DT web application and using case studies to investigate the result. The proposed framework received as input the original data model from a DT system that was implemented in a native application for desktop computers. This input was used in the DT data model design methodology proposed.

The native DT system enabled users to create digital replicas of physical systems and to simulate the systems' performance. This DT system could not, as many studies have shown from Chapter 1, be used on devices such as smartphones. This was due to the DT system being implemented as a native application and it requires too much computing power to execute.

Thus, there was an opportunity to create a cross-platform DT system which enabled access for many users to some of the functionalities that were present in the native application. Functionalities such as 3D model rendering, 3D model interaction and movement, and metadata access combined with data processing for executing calculations from the metadata were implemented in the cross-platform system. Simulation functionalities were deemed out of scope for the cross-platform system.

3.2 Web application optimised for DT functionalities

The native application's DT data model was used as input for the first method in the framework, namely, the design methodology for DT data models to create a data model that stores only the required information in the smallest amount of data possible.

Following this, the second method in the framework, namely, the development method for DT web applications was applied. This method is focused on enabling the design, implementation, and optimisation of DT functionalities for web applications that enable cross-platform access. This method was also used to enables the modular and scalable implementation of these functionalities.

The results of using the proposed framework to create a DT web application is shown in Figures 3.1, 3.2, and 3.3 in the following section.

3.2.1 Cross-platform access

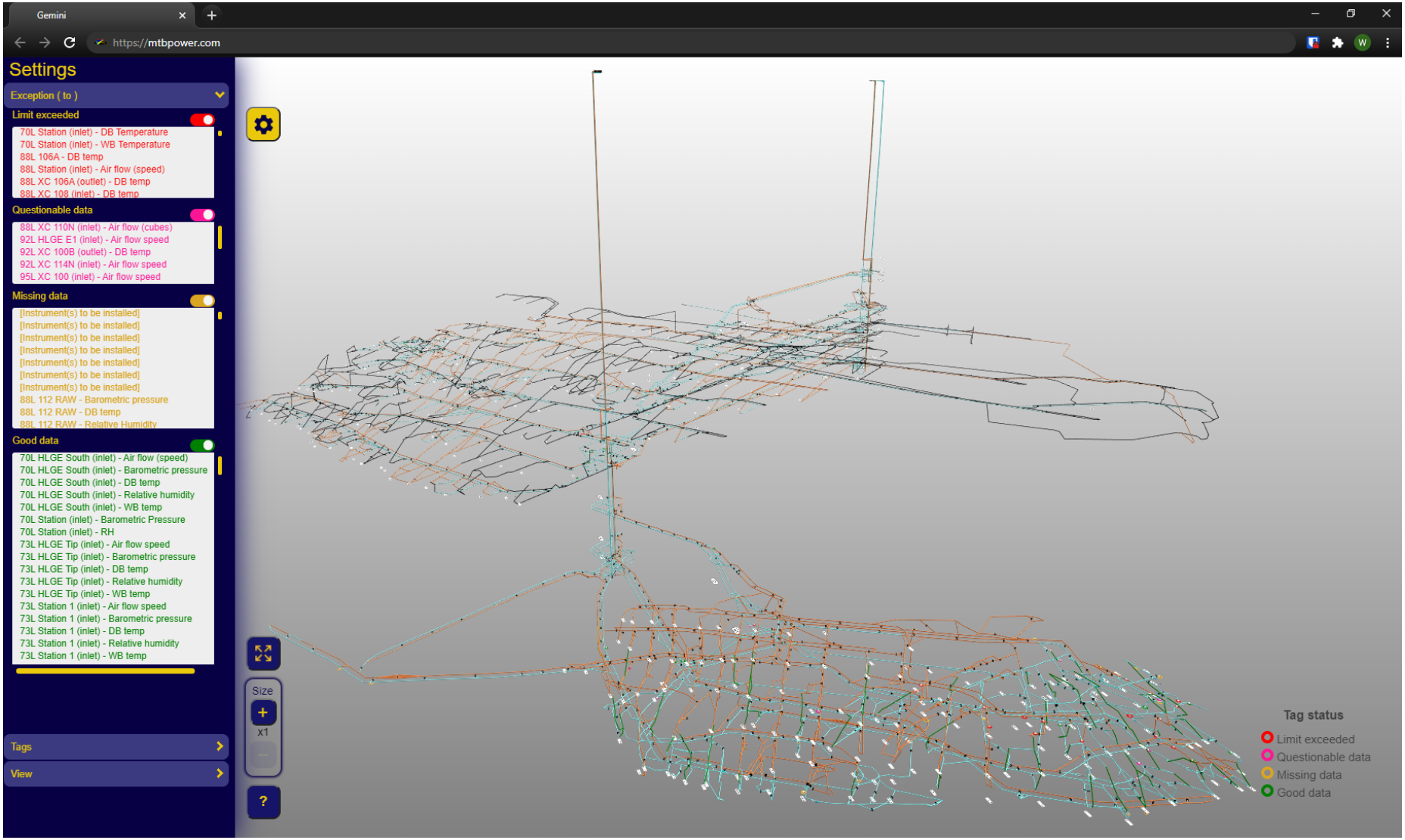


Figure 3.1: High-level overview of physical system on DT web application.

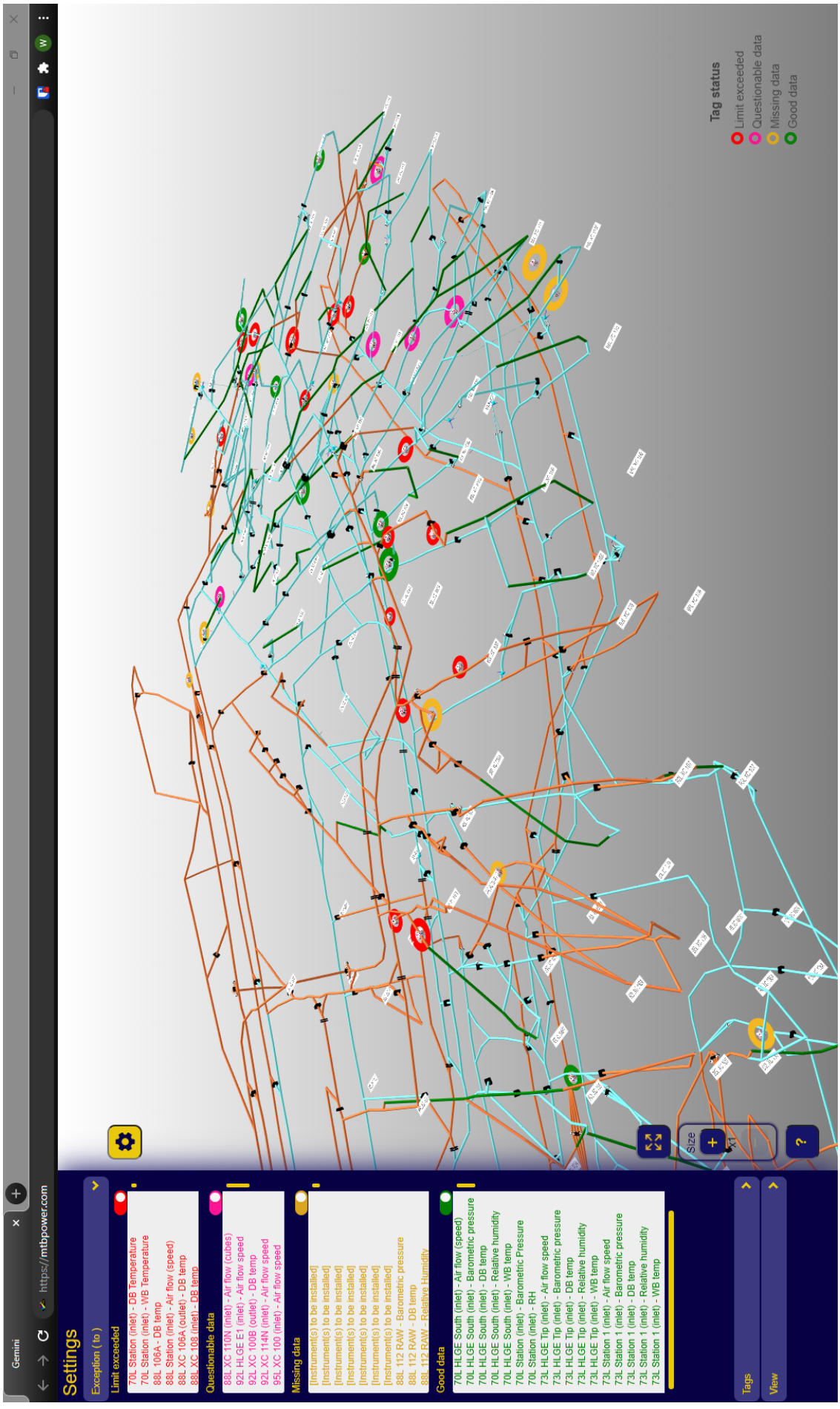


Figure 3.2: Intermediate-level overview of physical system on DT web application.

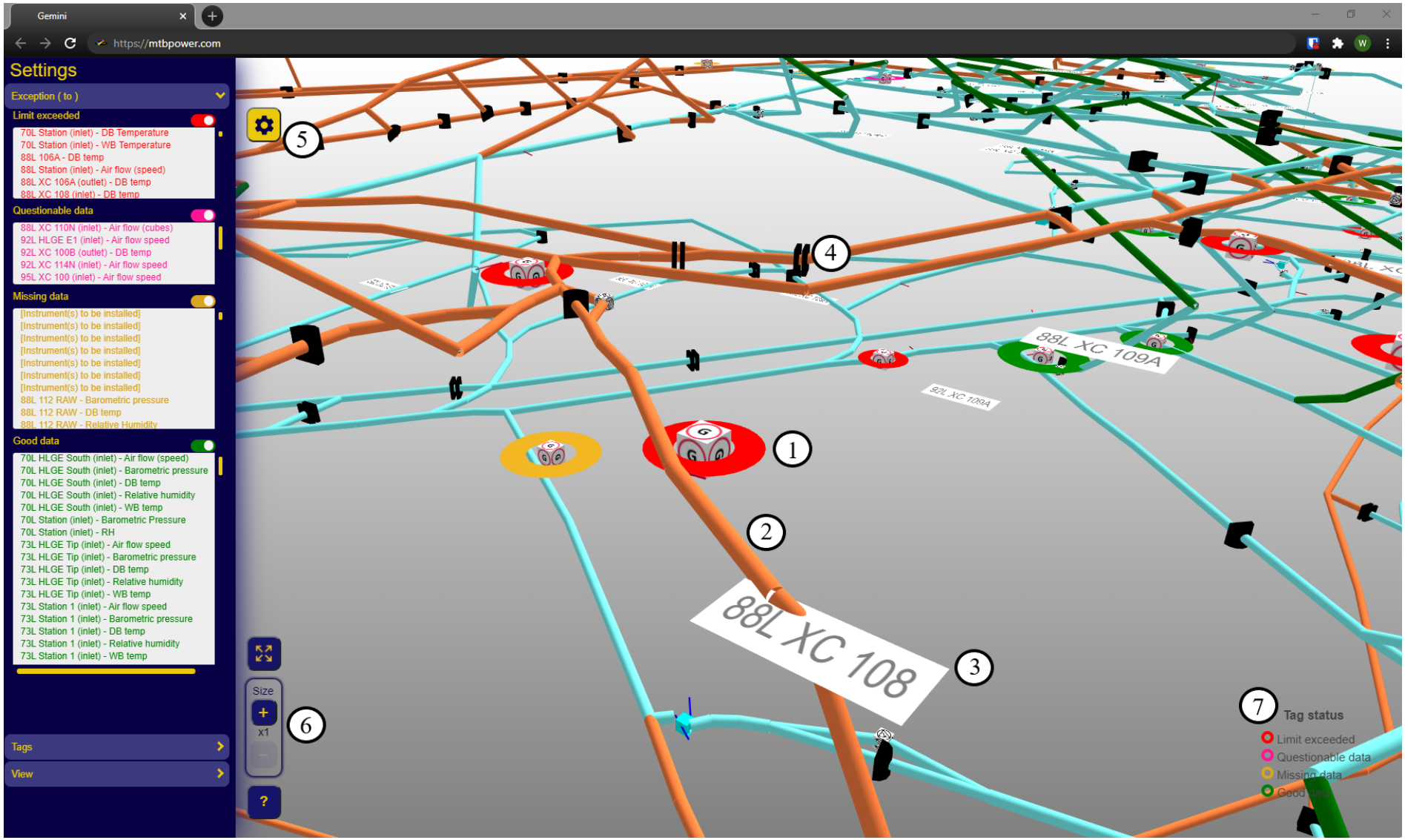


Figure 3.3: Low-level overview of physical system on DT web application.

DT web application overview

Figure 3.1 shows the result when a user accesses the DT web application. The web browser navigation bar was included to show that the DT system was implemented in a web application.

Many different web development libraries built for WebGL were investigated. Three.js was chosen for this application based on documented performance analysis as well as high quality documentation and examples. The library was initially released eleven years before this study took place and it is still actively maintained and new functionalities are also being developed.

Three.js abstracts some of the low-level implementation of WebGL into a library of typically used classes and methods which have been optimised to increase the performance of web applications requiring 3D model rendering and animations.

The web application rendered (in the area with the grey background) the digital representation of the chosen physical system (consisting of compressed air pipes, components, sensors, etc.) as well as custom text boxes (small plane objects) used for descriptions of areas or components. This physical system is one of the case studies used in this chapter.

The three case studies used in this chapter were chosen due to the difference in system complexity. Case study 1, shown in Figure A.1 in Appendix 1, is the least complex system of the three. The complexity of the systems is measured by the number of components, the number of detailed geometries, and the amount of metadata to be processed in the DT functionalities.

Case study 2, shown in Figures 3.1, 3.2, and 3.3 is more complex than case study 1 due to more components and metadata present. Case study 3, shown in Figures A.2, A.3, A.4, and A.5 in Appendix 1 is the most complex of all three as it contains the most components, intricate geometries, and the most metadata. The purpose of choosing increasingly complex case studies is to determine the scalability of the implemented DT functionalities in the web application. The figures of case study 1 and 3 are available in Appendix 1.

An existing API backend system (used for other systems besides the DT web application), which was hosted in a cloud environment, was extended with new functionalities for the DT web application. These functionalities were specifically related to the sensor data processing.

Table 3.1 below lists and describes each of the labels shown next to the right of the interface features of the DT web application in Figure 3.3.

Table 3.1: Visual components of the DT web application

Label	Name	Description
1	Sensor and exception ring	Represents a sensor on a component at that location in the system and describes its performance status.
2	Connection	In this case study, models a compressed air pipe between two points in the system.
3	Text box	User-customisable descriptions of regions within the system.
4	Custom geometry	Object for which a detailed geometry was rendered.
5	Sidebar controls	Collapses or shows the sidebar.
6	View reset, scale setting, and controls information	Resets the view frustum, changes the scale of the connections, and displays control instructions for users.
7	Exception legend	Describes the meaning of each type of exception ring.

The first HMI functionality discussed is the view frustum controls. This functionality enables the user to change their perspective of the digital system.

View frustum controls

The viewing perspective interactions between the user and the 3D rendered digital representation are the orbit controls, i.e. how the user rotates or moves the camera perspective around the 3D system. Orbit controls from the web development library were used to enable this interaction. Figure 3.4 visualises the orbit controls and the user's viewing frustum i.e., perspective.

The orbit controls enable the user to rotate the 3D system by allowing them to move their viewing frustum around on the sphere. This is done by holding down the right mouse button or using two fingers to touch and moving around. For devices without touch screens, e.g. typical desktop computers, the mouse controls are relevant. For devices with touch screens, the finger descriptions are relevant.

The point of origin of the orbit controls ($< 0, 0, 0 >$ in Figure 3.4) is the object which is currently focused on in the 3D system. When the DT system is first accessed, the default focus is set to the middle of the 3D system.

When a user wants to change the distance between the point of origin and their frustum, they can either use the middle mouse button to scroll or pinch two fingers on the screen. If a user does not want to move their perspective on the sphere, but rather on the plane which is orthogonal to their current focus point, then they can use the left mouse button or a single

finger to pan.

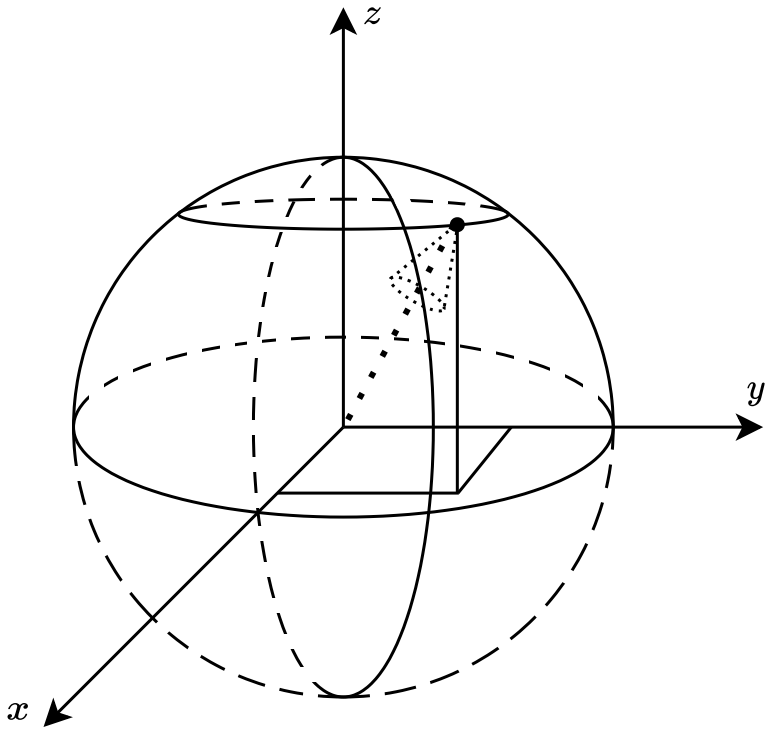


Figure 3.4: Orbit controls and viewing frustum visualisation.

These three different methods of rotating, moving, and changing the distance from the point of origin were already implemented in the web development library. The next functionality for 3D system interaction was implemented in the DT web application.

The functionality added enabled users to use a mouse or one finger to click or touch on one of the objects within the system. When this occurs, the newly-implemented functions change the focus point of the orbit control to the object which was clicked or touched on.

This functionality can be visualised from the difference of the frustum between Figure 3.2 and Figure 3.3. If a single click or touch occurred on one of the components in Figure 3.2, then the new functionality determines which component was clicked or touched and shifts the focus of the viewing frustum and reduces the length to the component (zooms in on the component) to the perspective shown in Figure 3.2.

This enables users to quickly change from a high-level overview of the entire system to a low-level view of a component which, e.g. is shown to be problematic (missing data from the sensor or erratic component performance). This brings up one of the most important functionalities of the DT web application which is to analyse component performance and visually display the results to the user of the DT web application.

Component status calculation and visualisation

The component performance analysis is based on a predefined set of ranges for the sensor values. This data processing occurs in the API backend system.

When the API receives a request from one of the instances of the DT web application for a status check on a system's components, it queries the data from a database (which is also hosted in a cloud environment), analyses the sensor data and limits provided, and sends the answers back to the DT web application.

Table 3.2 describes the different type of limits that can be set for these status checks. The values referred to in the table against which the limits are tested, are the sensor's data values.

Table 3.2: Limit types and descriptions

Limit type	Description	Exception cause
Lower	Single static value	Sensor values below limit values
Upper	Single static value	Sensor values above limit values
Profile lower	24 or 48 length profile	Sensor values below limit values
Profile upper	24 or 48 length profile	Sensor values above limit values
Calendar profile	An upper and lower 24 or 48 length profiles unique for each month of the year and day of the week	Sensor values above and below limit values respectively
Simulated lower	24 or 48 length profile generated by the desktop simulation application	Sensor values below limit values
Simulated upper	24 or 48 length profile generated by the desktop simulation application	Sensor values above limit values
SCADA tag lower	Actual sensor values stored in the database	Sensor values below limit values
SCADA tag upper	Actual sensor values stored in the database	Sensor values above limit values
SCADA calculation lower	The result from a calculation using actual sensor values stored in the database	Sensor values below limit values
SCADA calculation upper	The result from a calculation using actual sensor values stored in the database	Sensor values above limit values

The web application then creates the applicable exception rings as shown in Figure 3.3 (red,

yellow, pink, and green rings around the sensors). When these rings are created, an animation sequence is also created. This animation enlarges the x and y scale of the ring geometry to double their size over a set period of time, and then afterwards, back to their original size over the same period of time.

This creates a pulsing effect on the components in order to inform the user of components with problematic performance. The user must then act on the information provided by the DT web application. Autonomous control of the physical system from the DT web application was not implemented. The next category of functionalities discussed are those present in the blue Settings sidebar of Figure 3.1.

Sidebar functionalities

Figures 3.5a and 3.5b show the exception list and tag list of the sidebar respectively. The sidebar functions as a place to summarise the results of the status checks as well as provides visual filters and dataset customisation. In Figure 3.5a, the exception list is shown. This shows all of the systems statuses grouped into types of status.

When one of the entries of the list is clicked on or touched, then the focus of the view frustum shifts to the component on which the sensor/calculation is linked. The distance between the view frustum and the component is also updated to bring the component into focus.

This assists users to quickly identify where each of the sensors are and helps them to navigate to components of interest on a single click. This exception lists have a coloured slider to their top right. This slider sets which of the exceptions rings are visible.

This helps filter out exceptions in a large system and to avoid an overload of information. Providing too much information may cause the important information to get lost or ignored. The lists are sorted alphanumerically to assist searching for specific sensors or calculations in a large system.

In Figure 3.5b, the tags list is shown. This is a combination of all the sensors' and calculations' status into one list. The sliders at the top of the list are filters for the list and for users who only want to look at, e.g. calculations with limits linked to them.

A single click on entries in this list executes the same functionality as for the exceptions list, by zooming to (lerping) the component on which the sensors or calculation is linked.

The last section of the sidebar is where the dataset (from-date, to-date, and interval) is set as well as the selected layer, selected overlay, documents toggle, plane background toggle, and

schematic toggle as shown in Figure 3.6.

(a) Exceptions area. (b) Tag area.

Figure 3.5: Sensor and calculation data processing results.

The system is broken into layers that usually represent different levels of the deep-level mines. The user can select only one layer to be visible, or all of them. The overlays are additional

connection colours that may convey meanings such as planned connections, or connections that will be removed to increase efficiency.

The documents toggle, when 'on', shows white exceptions rings around components which have metadata linked to them, e.g. a PDF document for component maintenance management. The background toggle converts the custom text box background from white to transparent and keeps them orthogonal to the view frustum at all times.

The schematic button (the last button) toggles whether or not the DXF of the system should be shown to the user. DXF is a widely used format created by Autodesk for the use of models from AutoCAD in many other CAD tools and 3D rendering programs.

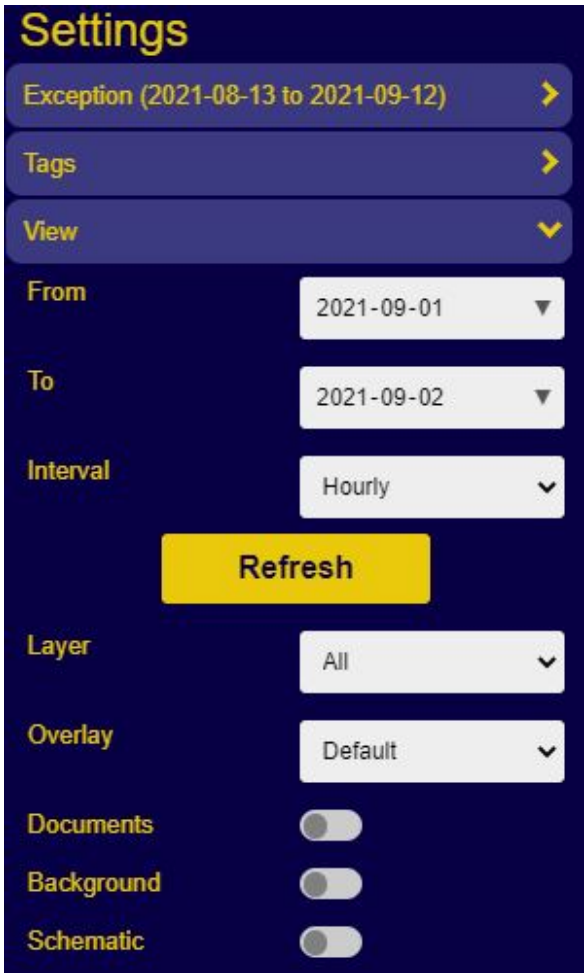


Figure 3.6: Date range, interval, and various other settings.

All the functionality in the sidebar is aimed at improving effective user interaction and the conveyance of important information from the DT system to the user besides only the 3D model rendering and exception rings.

Metadata access functionalities

Besides the status checks (in the form of pulsating exception rings), the DT web application also has metadata access functionalities. In Figure 3.7, the modal, which is shown when a component in the system is double-clicked or tapped on, is shown.

There are three tabs in the modal, namely, Scada, references, and general. The Scada tab contains the sensors (Scada tags[1]) and calculations linked to the relevant component. The result when any of these entries are clicked on or touched, is shown in Figure 3.8.

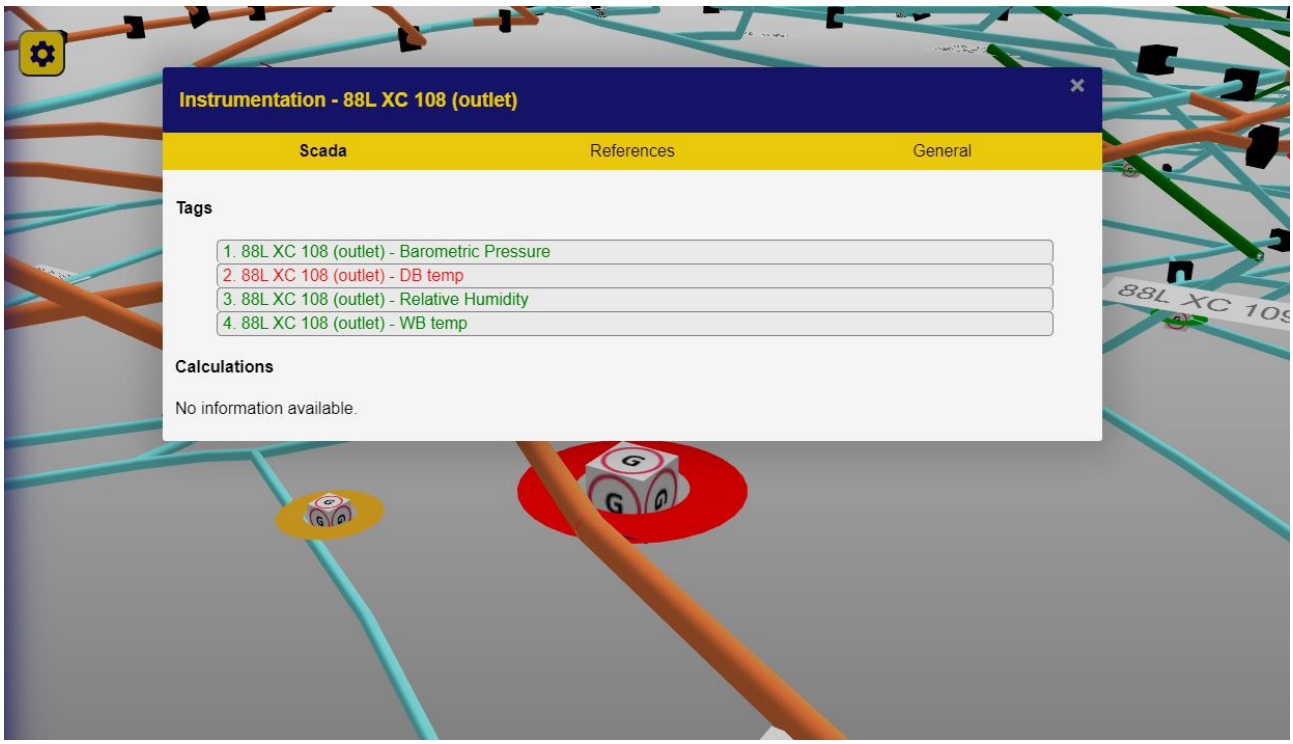


Figure 3.7: Sensors (tags) on and calculations linked to a component.

Figure 3.8 shows the graph which is displayed to the user on a Scada entry click or touch. The dataset for this specific graph can be altered and refreshed to inspect a different dataset. This does not affect the sidebar dataset.

These graphs enable the user to inspect at what time the limits were exceeded, the severity of the exception, as well as the average profile over a certain period. This helps users to effectively identify when and to what extent the sensors' values are exceeding the set ranges.

[1]Scada tags are database references to real-world sensors, they are the link between the sensor, the database, and the DT system

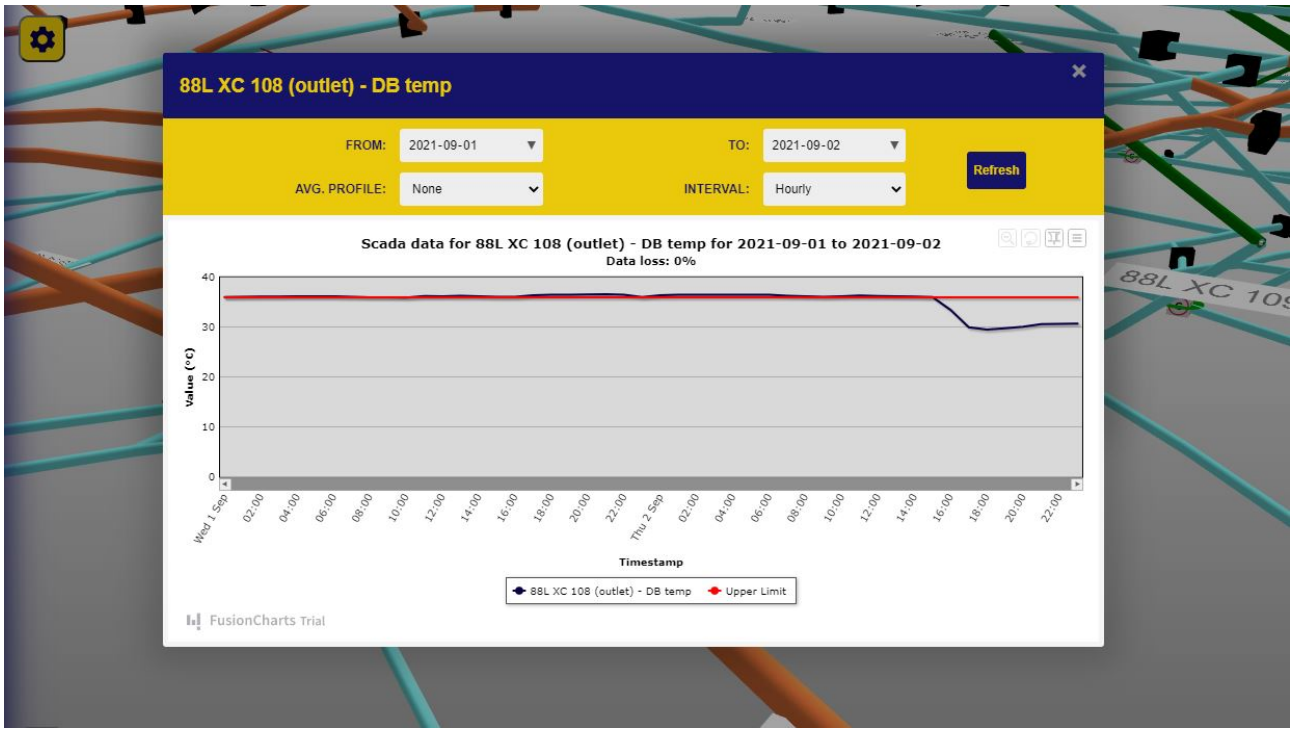


Figure 3.8: Sensor data up to the last half hourly interval.

The next tab in the modal is the references tab. Metadata such as PDF documents can be linked here. Figure 3.9 shows the list of documents.

Figure 3.9: Metadata (references) for a component.

These entries can also be clicked on to show their content, as shown in Figure 3.10. Some information from the document has been redacted for privacy reasons.

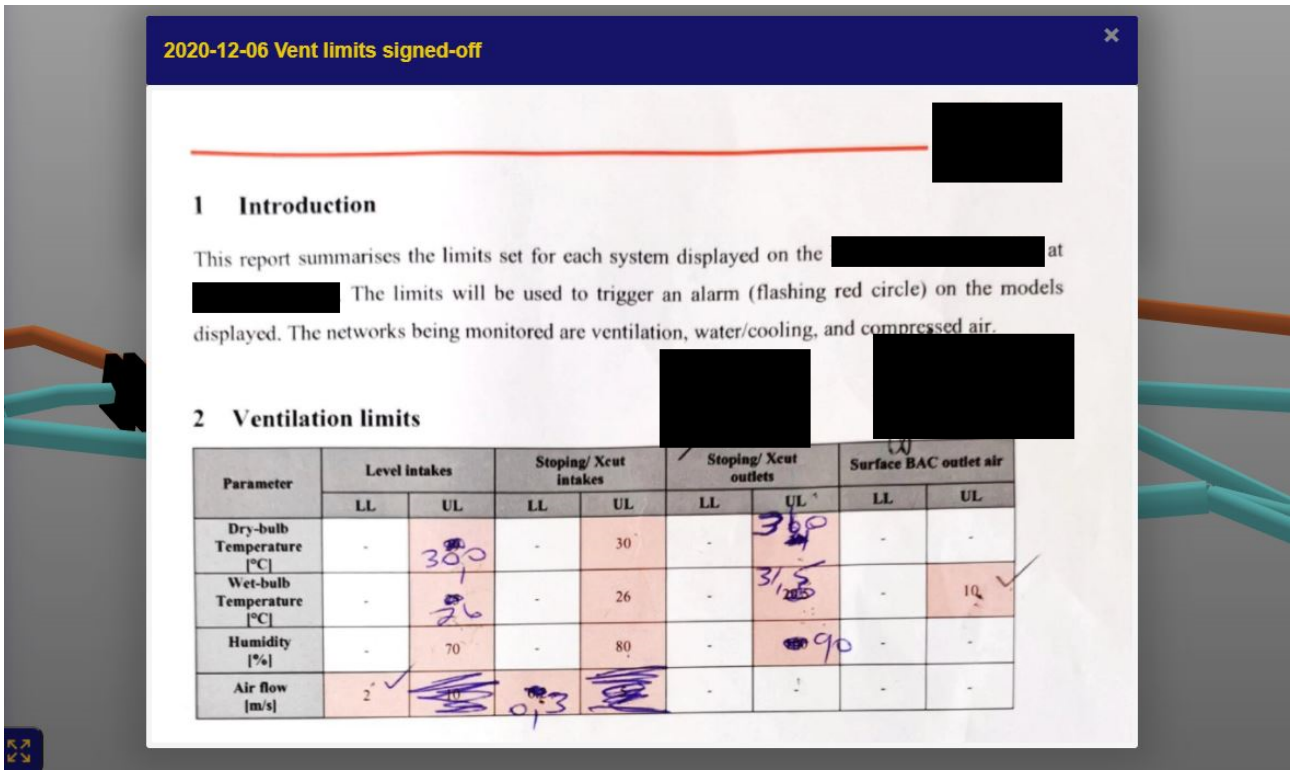


Figure 3.10: Linked reference with redactions.

This enables the user to inspect relevant documentation, which may consist of a maintenance plan. Finally, the general tab is simply another place to link more metadata. The users who configure these DT models can use these two tabs to store any documentation which is relevant to the component.

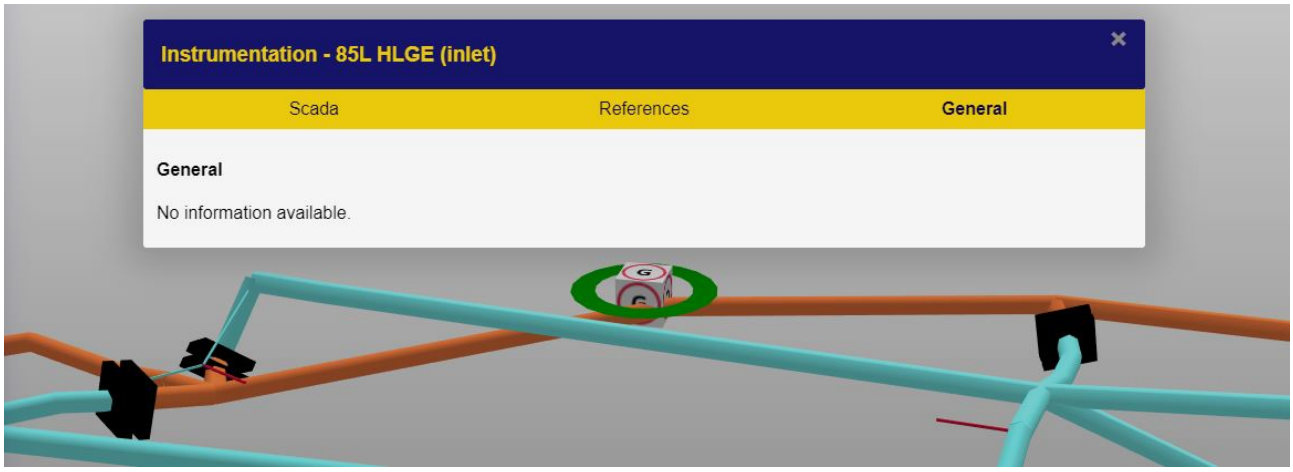


Figure 3.11: Other metadata types linked to the component.

3.2.2 Projects translated to the web application

Personnel using the native desktop DT application to simulate the systems have the option to publish the DT to the web application if they want to use its functionalities. One of the main reasons for doing this is cross-platform access to DT functionalities.

This cross-platform aspect of the DT web application enables access for many more users to the DT functionalities described in Subsection 3.2.1. This means that clients and personnel can access many different DT models representing complex systems with the purpose of inspecting performance and being alerted of system faults from their smartphones. This may lead to faster response times (e.g. the replacement of a faulty valve before any other components are damaged due to the fault), which may reduce lost production time and improve employee safety [27].

Many of these users may not have access to a computer due to the nature of their work, thus, the only way to enable consistent access to the DT functionalities for them is using the cross platform web application. The increase in number of projects published to the DT web application is shown in Figure 3.12.

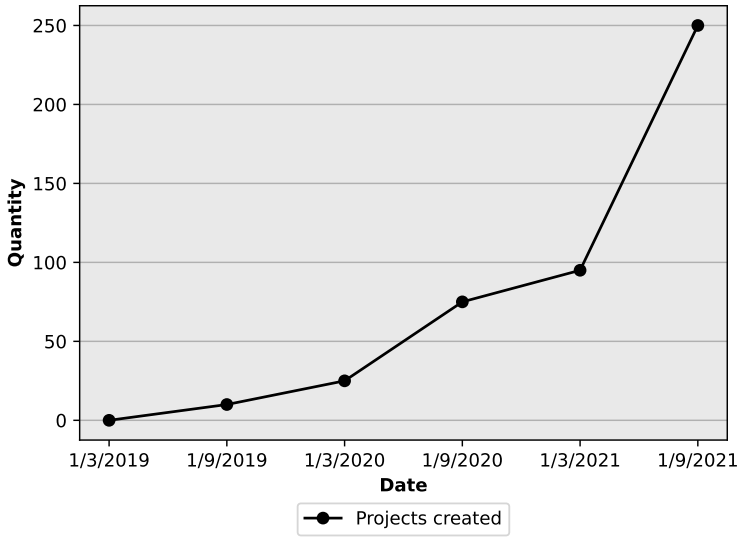


Figure 3.12: Number of projects created for the web application.

The number of projects since the start of development of the DT web application has shown consistent increase. Each of the large spikes were due to significant new functionality or performance increase (which will be discussed in Section 3.3).

3.2.3 User utilisation of the web application

The aim of the DT web application is to enable access to many users to DT functionalities. Figure 3.13 shows the increase in number of users over the development period of the DT web application.

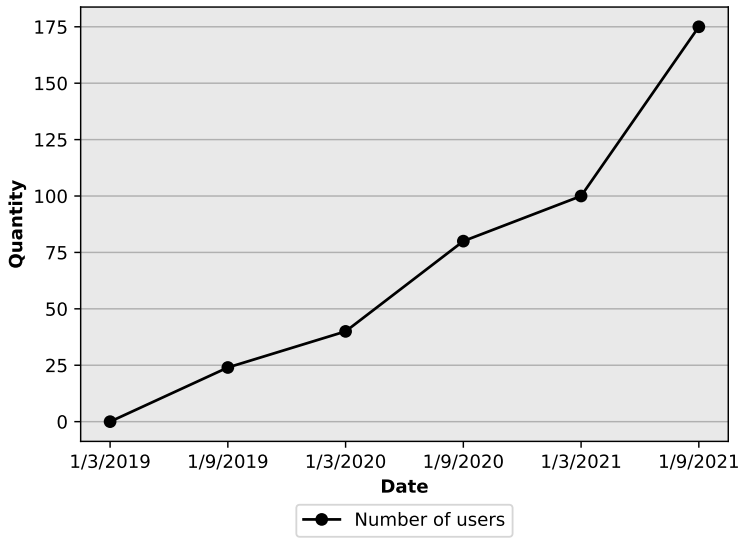


Figure 3.13: Number of users actively accessing the DT functionalities through the web application.

It can be seen that the large spikes in users coincide with the spikes in the number of projects published to the DT web application. The increase in the number of users can also, however, be greatly influenced by advertising done for the functionality.

3.3 Performance improvements

3.3.1 Preamble

Three case studies were chosen to measure the performance improvement of the system created using the framework proposed in the previous chapter. The first case study consisted of a relatively small system, while the second consisted of a more intermediate sized system. The final case study is the most complex of the three. Appendix 1 and Figures 3.1, 3.2, and 3.3 shows the three different case studies used.

The varying complexity in the case studies serves to show the scalability of the implemented functionalities. The level of scalability of the functionalities can be determined by analysing performance indicators of the implemented web application.

Two major performance indicators were used to measure the effectiveness of the framework in reducing the required computing power to access the DT functionalities implemented in the web application.

The first performance indicator is the required DT data model size. The larger the data model

size, the higher the required computing power from the device used to access the functionalities through the web application and the longer the download time and cost.

3.3.2 Reduction in required DT data model size

Figure 3.14 shows the DT data model size before and after the proposed data modelling method was applied to it. The average reduction of the DT data model size using the proposed method is 86.5%.

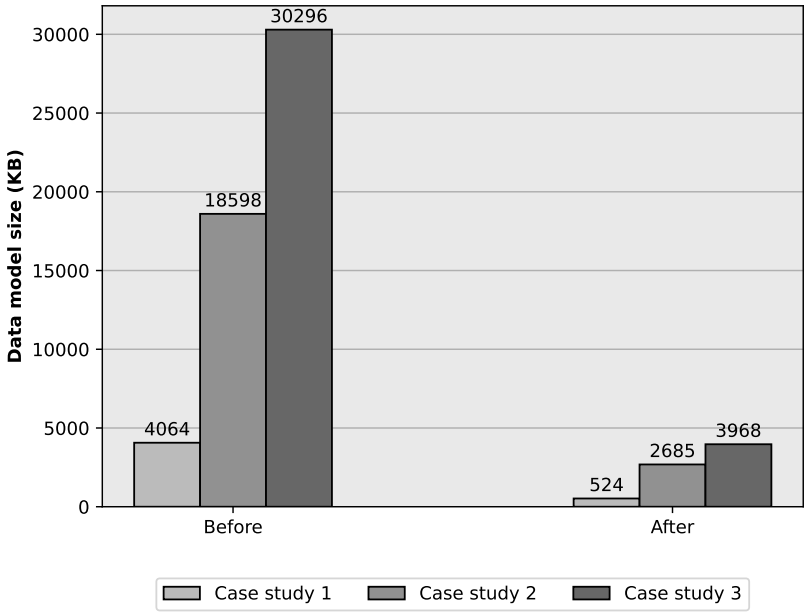


Figure 3.14: Data model size before and after the data modelling method was applied.

This is a significant improvement as the amount of data required to transfer the DT data model over a mobile network has been reduced by more than 86% on average for these three case studies. This also greatly reduces the amount of time required to transfer the DT data model over a mobile network.

This significant size reduction translates directly to greatly reducing the required computing power to access the DT functionalities on the web application. The second performance indicator considered is the framerate improvement over time as the proposed framework is used to implement and optimise the DT functionalities.

3.3.3 Framerate improvements

Framerate is the number of frames (images on the screen) a device can render and display to a user per second. It describes how smoothly the device display animations, e.g. rotating a 3D model of a system according to user input. The higher the frames per second (FPS), the smoother the animation sequence shown to the user. The average human being can see between

30 to 60 FPS, as shown in the study by Potter et al. [46].

Therefore, to achieve 60 FPS for devices such as smartphone, the proposed framework was applied. The resulting change in framerate for the three case studies over a period of eighteen months is shown in Figure 3.15.

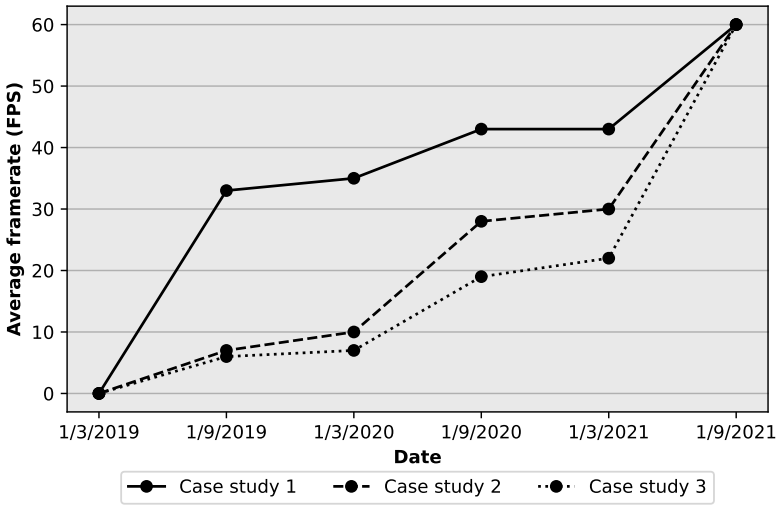


Figure 3.15: Average framerate improvement over the development period.

The proposed optimisation process to improve the framerate is a process applied to the software developed to render and display the DT models in the web application. Therefore, the process is not specific to each case study, it is a generic process to improve the performance of the rendering software used to create and animate the DT models in the web application.

On 1 March 2019, the system had not been implemented, and hence, 0 FPS was achieved. On 1 September 2019, the initial implementation was done, and the baseline framerates were recorded. Case study 2 and 3 were showing much lower than the required framerates with both being below 10 FPS.

The next period stopping at 1 March 2020 showed only a slight increase in FPS for all three case studies, as more functionalities were added and focus was not on optimisation. The results at 1 September 2020 showed a significant increase in FPS. This was due to less functionalities being added and more time spent on the optimisation of the existing functionalities.

The results shown for 1 March 2021 only indicated a slight increase in performance for case studies 2 and 3. This was again due to the fact that more functionalities were required, and therefore, not much focus could be placed on optimisation.

The last timestamp, 1 September 2021, shows 60 FPS for all three case studies. This was due to important optimisations that took place between 1 March 2021 and 1 September 2021. This shows that the system is scalable as the implemented web application achieved 60 FPS

even for complex systems such as case study 3.

These results were obtained by accessing the implemented DT functionalities on a Huawei P20 Pro 2018 smartphone. The device specifications are listed below.

- CPU: Octa-core (4×2.4 GHz Cortex-A73 and 4×1.8 GHz Cortex-A53)
- RAM: 6 GB
- Screen resolution: 2240×1080 pixels

3.4 Validation

Validation is a low-level analysis of the methodologies proposed in order to determine whether or not they achieve the objectives of the study. In this section, the two proposed methodologies will be validated.

3.4.1 Design methodology for Digital Twin data models

The objective for this method, as stated in Section 1.7, was to create a data modelling method to take existing DT data and create a data model which has been designed and optimised for use on devices such as smartphones.

The criteria for data models designed and optimised for devices such as smartphones are that the data model size must be feasible for data transfer latency and cost implications. The size also affects the performance of possible systems using the data model. Finally, the format must be supported on many different types of devices, such as smartphones, laptops, and desktop computers.

The data model format chosen was the JSON format, as discussed in Subsection 2.2.3, which is a widely supported data format. This was used for the DT data model format.

The data model size and complexity criteria was also addressed by the method as there are steps in the method specifically aimed at reducing the complexity and size of the data model, as discussed in Subsection 2.2.3 and 2.2.4. These steps include starting at the lowest level of detail (LOD) and iterating to higher LODs to satisfy the data model requirements, and also replacing topological data features with relational trees.

Although this is static validation, examples within each of the subsections mentioned show

and discuss the example results from the method. It has been validated from the discussion above and the examples in Section 1.7 that the first objective of this study was achieved.

3.4.2 Development method for Digital Twin web applications

The functionalities in DT systems can be broken down into two main categories, namely, the HMI functionalities and the data processing functionalities. The validation of the implementation process for DT web applications, which is the first process in the methodology, applied to HMI functionalities is discussed first.

The process of validation of the first category consists of, firstly, a manual process of testing the functionalities, and secondly, a peer review process analysing the implementation and also testing the functionalities. The first process is described in the steps below.

1. Setup a test environment that, as closely as possible, mimics the environment in which the final DT system will execute,
2. Create steps with which the required functionality can be tested (steps which a user will take to use the functionality), ensuring as many edge cases are included as possible,
3. Iterate through the steps and ensure the results are as expected,
4. Apply this to many different case studies (different DTs with varying complexity) in order to determine the performance of the functionality as the systems scale and increase in complexity.

The DT functionalities used to validate the method is described in Table 3.3 below. It also shows the result of applying the testing process described above.

Table 3.3: Validation of the implementation process for DT web applications on HMI functionalities

DT HMI functionality	Result
Render 3D digital model of the physical system in a web application	✓
Enable 3D rotation of the model through touch or mouse input	✓
Lerp the camera to a component on appropriate user input	✓
Display animations to highlight components performing poorly	✓

The results from rendering the 3D model, enabling 3D rotation of the model, lerping of the camera, and lastly, the display of sensor data were all positive using the testing process described. Therefore, the implementation process applied to HMI functionalities has been

validated.

The second and last category of DT functionalities is the data processing functionalities. This consists of processing such as sensor data aggregation, calculations using multiple different sensors' data, and the identification of components performing poorly based on a set of ranges of expected sensor data values.

The way these functionalities are validated is through the use of automated integration tests (ITs) shown in Figure 3.16. These ITs are functions that receive as input a set of fixed data specifically created to test each functionality. The data must be valid in terms of the type of data the functionality receives to process the result.

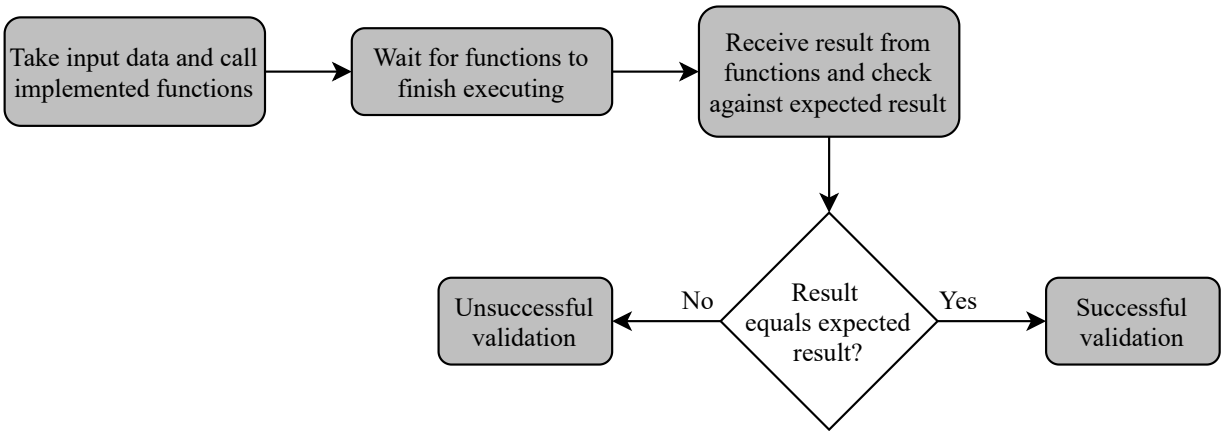


Figure 3.16: Automated ITs used for validation of proposed method on data processing functionality.

The IT then send this set of test data to the functions that were implemented for the functionality. The results obtained from the functions are then tested against a set of data describing exactly what the expected results are.

If any of the results obtained from the functions differ from the expected result, then there is an issue which must be resolved. Table 3.4 shows the results of the implementation of the example data processing functionalities already discussed.

Table 3.4: Validation of the implementation process for DT web applications on data processing functionalities

DT data processing functionality	Result
Sensor data fetch from database, aggregation, and representation in graphs	✓
Calculations using sensor data	✓
Identification of components with faulty or problematic performance	✓
Metadata fetch from database, e.g. documentation related to components	✓

All of the results in the table are positive, and hence, the implementation process applied to the second and last category of DT functionalities has been validated. The implementation process has been validated for both the categories of DT functionalities and therefore, the implementation process has been validated.

The last process of the development method to validate is the continuous development and maintenance process which, as discussed in Subsection 2.3.6, aims to enable the maintenance and optimisation of the DT functionalities over time. It is based on the PDCA cycle.

Performance profiling tools for web applications will be used for the validation of this part of the methodology. These tools enable the inspection of many different device execution parameters, such as the amount of device memory used to render and enable HMI with the DT web application.

Profiling tools also enable the inspection of the number of triangles and geometries required to render the model. Finally, the amount of time that functions take to execute can also be measured, e.g. a single render cycle execution time.

Table 3.5 shows the reduction of these execution parameters over time as the continuous development and maintenance process was applied to a case study. The table shows that for each functionality the process was applied to, an improvement of the relevant execution parameter can be seen after each iteration.

Table 3.5: Validation of the continuous development and maintenance process

Functionality	Execution parameter	Initial	Iteration 1	Iteration 2
Render cycle	Average execution time	28ms	20ms	18ms
Data model decompression	Average execution time	10s	7s	3.6s
3D model rendering	Memory usage	3.2GB	2.1GB	700MB
Number of reported issues	Errors	14	10	7

The values in Table 3.5 were obtained by accessing the implemented DT functionalities on a Lenovo Legion Y540-17IRH laptop and evaluating the chosen execution parameters while applying the proposed continuous development and maintenance process. The device specifications are listed below.

- CPU: Intel Core-i5 9300H (2.4 GHz, 4 cores, and 8 logical processors)
- RAM: 16 GB
- Screen resolution: 1920×720 pixels

Memory usage and execution time are both important parameters to inspect, as they greatly affect the scalability of DT systems. DT systems become more scalable as less memory is being used and as less time is spent executing functionality. This is because less memory usage means that devices with relatively low memory compared to desktop computers, e.g. smartphones, are then adequate to enable access to the DT functionalities.

A reduction in the required execution times decreases the amount of time users must wait in order to access the functionalities. It may also increase the FPS which can be rendered and shown to the user.

The results from the case studies of various functionalities indicate that the continuous development and maintenance process which forms part of the overall development method has been validated. Hence, the development method for DT web applications has been validated to correctly achieve the objectives created.

Both of the methods proposed have been shown to correctly achieve the objectives that were created in Section 1.7. The results of using the proposed framework to create a DT web application were shown in Sections 3.2 and 3.3. The following section consists of the discussion of the results obtained.

3.5 Discussion of results

The functionalities and performance achieved were the results of applying the framework proposed in Chapter 2 to create a web application to enable access to many users to DT functionalities. Three case studies of deep-level mines were used to measure performance indicators of the implemented functionalities.

The web application implementation results will be discussed first, and following that, the performance improvement results will be discussed. Figure 3.1 shows the content which is displayed to the user when opening the DT web application.

The content consists of the 3D model of the physical system contained within the area with the grey background, as shown in Figure 3.1. In this case, it is the digital representation of the compressed air network of a deep-level mine case study. The blue sidebar on the left of the 3D model contains the limit exceptions data, sensor filters, as well as dataset settings in the dropdown lists.

The first functionality implemented for the web application was the rendering of the 3D digital model of the physical system, and following this, the camera controls were

implemented to enable users to move around and through the 3D model. This allows users to change their perspective from a high-level view of the entire system, to a low-level view of components within the system, as shown from the difference between Figures 3.2 and 3.3.

This functionality shows one of the advantages of using a DT to represent a physical system, namely, a high-level overview of the entire system is possible, as well as a low-level view of a single component's performance. Automated exception calculations (limit tests) was the other major functionality implemented for the web application. The API backend was used to execute the processing intensive exception calculations. Once the web application received the results from the API, pulsing rings are rendered around the relevant components.

The API backend was chosen to execute the intensive data processing to reduce the required processing power of the device on which the web application was accessed. There is a waiting period for the web application to request the answer and wait for the API to calculate it, however, the web application's existing functionality can be used during this waiting period without reduction in performance as the device's processing power is not being used to calculate the answer.

It was not part of the DT system requirements to be a live system (i.e. real-time). Therefore, this short waiting period was deemed acceptable as using the API means that the device's computing power is not being used, which could negatively impact framerate performance of the DT functionalities.

Exceptions are upper and lower limits, as described in Table 3.2, which enable users to set static or dynamic limits on sensor data values. When the sensor data is above or below the respective limit, then the red exception rings are rendered around the applicable component in the 3D model.

These exception rings have a pulsing animation sequence which alerts the user that the sensor's data is out of the expected range, and the component's performance should be investigated. This functionality enables the automated testing of sensor data against static, e.g. a 48 point predefined profile, or dynamic data, e.g. other sensor data values.

These tests automate the process of the user inspecting all the components in the system by hand to determine whether their sensor data values are within the expected ranges. The automation of this process significantly speeds up fault finding within these large and complex systems.

Missing data and quality exceptions were also part of the exception functionality. Any sensors with missing data, or a missing ratio above a set amount, were flagged by yellow exception rings. Components with quality exceptions, e.g. sensor data values which should not be

possible, were flagged with pink exception rings.

The sidebar allowed further HMI by enabling users to quickly select exception entries, and then the camera would focus on the applicable component. This further increases the speed at which faulty components can be identified. Users must then act on the information given by the exception rings.

Sensor data graphs were also part of the implemented functionalities. This enables users to get a detailed view of when and for how long the sensor values exceeded the defined limits. Other metadata access was also part of the functionalities. This enables users to download documents related to the component.

All of these functionalities enables the user to utilise the DT web application to quickly identify problematic components and sensors within the system. It also allows users to get a high-level overview of the performance of the system.

The results obtained for the performance improvements also showed positive results. Three case studies were used, each of them increasingly complex to show the scalability of the web application. Two main performance indicators were considered for the three case studies, namely, the reduction in the required DT data model size, as well as framerate improvements over time.

The reduction in the required DT data model size is shown in Figure 3.14. It shows an average reduction of 86.5% across the three case studies. This shows that the data modelling method does significantly reduce the required DT data model size, and through doing this, reduces the amount of data that needs to be transferred over a mobile network.

The framerate improvement is shown in Figure 3.15. Some periods over the total 18 month period did not show a significant increase in FPS, however, during these periods many new functionalities were added. Hence, the focus was on new functionalities and not optimisation.

Some periods within the 18 month period, especially the period between 1 March 2021 and 1 September 2021, showed significant increases in FPS. This was due to less new functionality being required at that time, and hence, focus could be placed on the optimisation of the functionalities.

The increase in FPS was relative to the complexity of each case study, which shows that the functionality's scalability increased over time by applying the proposed framework. The result obtained for 1 September 2021 shows 60 FPS for all three case studies, which means that the required performance has been achieved for the three case studies. Even more complex systems should be used as case studies to further analyse and improve the functionalities in

the future.

The proposed framework was validated in Section 3.4. The validation was required to determine whether the proposed methods produced the correct results and to test whether the two methodologies proposed achieved the objectives of the studies.

The first methodology was validated by determining if the method produces a data model in a widely supported format. The data format used by the method is the JSON format, which is widely supported. The DT data model produced has also been validated as modular and scalable.

The second methodology was validated by determining whether the correct functionalities were implemented using the method, as shown in Table 3.3. Automated integration tests were also created, as shown in Figure 3.16, and used to test whether the data computations, such as the exception tests, produced the correct results. The results of the validation are shown in Table 3.4 and show that the method has been validated.

Finally, the validation of the continuous development and maintenance part of the development method was done. The results are shown in Table 3.5. The continuous development and maintenance process does improve the performance indicators over each iteration.

The results from using the proposed framework to create a web application enabling access to DT functionalities have been discussed. The proposed framework, and both of the methodologies within it, have been validated. The following section summarises the chapter and gives an overview of the objectives achieved by the proposed framework.

3.6 Conclusion

The first objective of this study, as stated in Section 1.7, is to create a data modelling method to take existing DT data and create a data model which has been designed and optimised for use on devices such as smartphones.

The criteria for data models designed and optimised for devices such as smartphones are that the data model size must be feasible for data transfer size, cost, and time implications. The size also affects the performance of possible systems using the data model. Finally, the format must be supported on many different types of devices, such as smartphones, laptops, and desktop computers.

The second objective of this study was to create a development method for DT web applications to provide access for many users to DT functionalities. These functionalities were required to be scalable and modular, so that the resulting DT web application is performant enough to enable many users to use these functionalities on complex systems.

The results shown and discussed throughout this chapter show that the two proposed methodologies have addressed the respective objectives of this study. The results show an average decrease in DT data model size of 86.5%, cross-platform DT functionality access, and performance improvement over time by using the framework.

Functionalities enabling access to metadata such as sensor data as well as data processing functionalities, e.g. exception calculations, were also implemented using the proposed methodology.

These functionalities, and the functionalities enabling the rapid change of perspective of a user through HMI, removes the need for users to manually inspect each component for problematic performance. It also provides a high-level overview of the entire system, while also enabling low-level inspection of component performance.

The increase in the DT system performance over time by applying the continuous development and maintenance process was also apparent in the results. The increase in framerate over time shows the significant increase in DT system performance as the method is applied.

The results were validated to ensure that the proposed methods produce the correct results. The validation showed that the correct results were achieved for the data modelling method as well as the development method for DT web applications.

The performance improvement results using the case studies showed that even for the most complex case study, the DT web application achieved the target of 60 FPS by using the proposed framework to optimise the DT data model and the required DT functionalities. The following chapter concludes this study and provides recommendation for future work.

4 Conclusion and recommendations for future work

4.1 Conclusion

Digital Twins (DTs) provide instant physical context to data in complex engineering systems such as deep-level mines. They assist with mining operations performance analysis, planning, and decision making when integrated with metadata, e.g. sensor data. However, these DTs are computationally intensive and typically require powerful desktop computers to be used instead of much less powerful mobile devices.

DT data models, DT reference architectures, and DT applications were investigated and studies relating to them were analysed. For the DT data models category, all the studies either focused on topological data models or geometric data models with additional metadata.

Almost all the studies considered and relating to DT data models were shown to place emphasis on DT reference architectures. All of these studies included integration with either topological or geometric data. However, less than half focused on sensor data integration into the data model. Furthermore, only two studies from the DT data model category focused on cross-platform access and scalable DT functionalities. Only one considered continuous development and maintenance.

A possible gap was identified for a DT data modelling method that focuses on scalable, cross-platform access to DT functionalities and that also focuses on continuous development and maintenance.

Studies focused on extending or proposing DT reference architectures were analysed next. These reference architectures include where and how the DT data model should be stored, how the different services within the overall DT system interact with each other, and how users interact with the DT system and its services.

All of these studies focused on sensor data integration, however, only two studies focused on cross-platform access, and, only one focused on scalable DT functionalities. The gap identified for the first research category correlated to the gap identified in the second category, namely that there was a need for a DT reference architecture and DT data modelling method which enables cross-platform access to scalable DT functionalities which also includes steps for continuous development and maintenance.

Finally, studies relating to DT applications were investigated. Many of the studies focused

on industrial applications, such as autonomous production and automated fault detection. Many of the studies also focused on simulation applications, such as optimisation models for production services and maintenance optimisation for multi-unit systems.

Almost all the studies focused on sensor data integration as it has been discussed that sensor data is critical for the calibration of simulation models. Approximately 50% these studies focused on DT reference architectures and scalable DT functionalities while none focused on cross-platform access and continuous development and maintenance.

The studies analysed show that DT systems provide a unique and powerful way to interact with and gain information pertaining to a physical system using a digital system. These digital systems enable easy accessibility to many previously mentioned services which lead to increased system time- and cost-efficiency.

However, no study in the categories of DT data models, DT reference architectures, and DT applications have focused on cross-platform, scalable access to DT functionalities while also focusing on continuous development and maintenance of the DT system.

Hence, a need existed to enable access for many users to DT functionalities for complex engineering systems using web applications and data models that have been optimised for low-powered devices, e.g. smartphones.

Two objectives for this study were proposed to address this need. Firstly, to create a data modelling method for DT data models optimised for low-powered devices. Secondly, to create a development methodology for DT web applications enabling scalable, cross-platform access to DT functionalities with additional focus on continuous development and maintenance.

The data modelling method for DT data models within the overall framework, was split into three parts, namely, the data model requirements, the data model design, and the data model optimisation.

The requirements enable the definition of what information was necessary for the DT data model. The data model design process enables the creation and updating of the DT data model in a modular and scalable manner.

The data model optimisation is aimed at reducing the complexity of the data model and reducing the data model size. All of these processes together, were proposed to enable the creation of a DT data model that contains all the necessary information in as small data model as possible.

The development methodology for DT web applications within the overall framework was split

into three parts, namely, designing a solution, implementing the functionality, and identifying lacking performance or functionality to improve and optimising the identified functionality.

The first part, designing the solution, was proposed to prevent the re-implementation of existing functionality, which wastes time and money. This process is also aimed at designing new functionality based on the scope of the system, which is also defined using this process.

The process to implement the functionality was proposed to enable the use of industry standards and the latest research for the functionality required. A modular and scalable implementation is also one of the main focuses of this process.

The continuous development and maintenance was proposed to ensure that the DT system remains up-to-date with the latest industry standards as well as the latest research and best practices. The inner loop in Figure 2.10 shows the PDCA cycle which the continuous development and maintenance process was based on.

The process is specifically aimed at reducing the likelihood of the system becoming a legacy system and, more importantly, ensuring that the system uses state-of-the-art methods for improved performance. This process was also proposed to create an opportunity for new research to be conducted and enable contribution to engineering knowledge.

The two methods proposed were verified using various case studies and experiments. The results from the verification show that the methods both correctly achieve the objectives for which they were proposed.

The verification process showed promising initial results from the two methodologies of the framework. The framework was applied to various case studies of different deep-level mines and the results were presented, analysed, and discussed.

The results shown and discussed show that the two proposed methodologies have addressed the respective objectives of this study. The results show a decrease in DT data model size of more than 86%, cross-platform DT functionality access, and performance improvement over time by using the framework.

Functionalities enabling access to metadata such as sensor data as well as data processing functionalities, e.g. exception calculations, were also implemented using the proposed methodology.

These functionalities, and the functionalities enabling the rapid change of perspective of a user through HMI, removes the need for users to manually inspect each component for problematic performance. It also provides a high-level overview of the entire system, while also enabling

low-level inspection of component performance.

The increase in the DT system performance over time by applying the continuous development and maintenance process was also apparent in the results. The increase in framerate over time shows the significant increase in DT system performance as the method is applied.

The results were also validated to ensure that the proposed methods produces the correct results. The validation showed that the correct results were achieved for the data modelling method as well as the development method for DT web applications.

The performance improvement results using the case studies showed that, even for the most complex case study, the DT web application achieved the target 60 FPS by using the proposed framework to optimise the DT data model and the required DT functionalities.

Due to the reduction in complexity and size of the data model as well as the performance of the web application, access to DT functionalities was enabled for many users with mobile devices such as smartphones. The framework created can be applied to DT data models in many different industries, such as mining and manufacturing industries, to enable remote access for users to DT functionalities.

4.2 Recommendations for future work

The DT web application implemented using the proposed framework did not include simulation functionalities. However, as the DT web application uses an API backend to perform data processing such as exception calculations, it would also be possible to perform simulation calculations in the backend. This would not affect the performance of the web application, as the API backend would be doing almost all of the work.

If the functionalities were extended even further to allow users to build the DT system in the web application, then the desktop simulation software would no longer be required. Users would be able to build, simulate, and monitor the DT system from this single cross-platform web application.

This study did not include qualitative feedback from the users to analyse ease-of-use of the web application. However, a qualitative study of the DT web application may show methods to improve the usability and user-friendliness of the web application.

The proposed framework can also be extended to include bidirectional data transfer, which, if combined with SCADA integration, would enable automatic control of the physical system by

the DT web application. Many possibilities arise from this, e.g. automated control strategies based on real-time data and real-time optimised simulations.

References

[1] R. Wagner, B. Schleich, B. Haefner, A. Kuhnle, S. Wartzack, and G. Lanza, "Challenges and potentials of Digital Twins and Industry 4.0 in product design and production for high performance products," *29th CIRP Design Conference*, vol. 84, pp. 88–93, 2019.

[2] S. Aheleroff, X. Xu, R. Y. Zhong, and Y. Lu, "Digital Twin as a service in Industry 4.0: an architecture reference model," *Advanced Engineering Informatics*, vol. 47, Jan 2021.

[3] L. Ortega, J. Jurado, J. Ruiz, and F. Feito, "Topological data models for virtual management of hidden facilities through digital reality," *IEEE Access*, vol. 8, pp. 62584–62600, 2020.

[4] D. D'Amico, J. Ekoyuncu, S. Addepalli, C. Smith, E. Keedwell, J. Sibson, and S. Penver, "Conceptual framework of a digital twin to evaluate the degradation status of complex engineering systems," *7th CIRP Global Web Conference*, vol. 86, pp. 61–67, 2019.

[5] R. Stark, C. Fresemann, and K. Lindow, "Development and operation of Digital Twins for technical systems and services," *CIRP Annals*, vol. 68, no. 1, pp. 129–132, 2019.

[6] F. Tao and M. Zhang, "Digital Twin shop-floor: A new shop-floor paradigm towards smart manufacturing," *IEEE Access*, vol. 5, pp. 20418–20427, 2017.

[7] R. Sahal, S. H. Alsamhi, J. G. Breslin, K. N. Brown, and M. I. Ali, "Digital Twins collaboration for automatic erratic operational data detection in Industry 4.0," *Applied Sciences*, vol. 11, no. 7, 2021.

[8] G. Wiro Sasmito and M. Nishom, "Development of web-based application in population administration system using scrum framework," *International Journal of Web Applications*, vol. 11, pp. 18–26, Dec 2019.

[9] D. Mourtzis, M. Doukas, F. Psarommatis, C. Giannoulis, and G. Michalos, "A web-based platform for mass customisation and personalisation," *CIRP Journal of Manufacturing Science and Technology*, vol. 7, no. 2, pp. 112–128, 2014.

[10] F. Toffalini, J. Sun, and M. Ochoa, "Practical static analysis of context leaks in Android applications," *Software: Practice and Experience*, vol. 49, Nov 2018.

[11] P. Ozhdikhin and D. Shkurko, "Creating and porting NDK-based Android applications for Intel architecture," *Intel Technology Journal*, vol. 18, no. 2, pp. 86–109, 2014.

[12] G. Lee, "Managing change with CAD and CAD/CAM," *IEEE Transactions on Engineering Management*, vol. 36, no. 3, pp. 227–233, 1989.

[13] W. Zeng, X.-J. Chang, and J. Lv, "Design of data model for urban transport GIS," *Journal of Geographic Information System*, vol. 2, pp. 106–112, 2010.

[14] Y. Ding, Y. Yang, C. Wu, H. Shao, and H. Li, "A multidimensional cadastral topological data model: design and implementation," *IEEE Access*, vol. 7, pp. 931–943, 2019.

[15] A. Schmetz, T. H. Lee, M. Hoeren, M. Berger, S. Ehret, D. Zontar, S.-H. Min, S.-H. Ahn, and C. Brecher, "Evaluation of Industry 4.0 data formats for Digital Twin of optical

components," *International Journal of Precision Engineering and Manufacturing-Green Technology*, vol. 7, pp. 573–584, Mar 2020.

[16] A. Borrmann, T. Kolbe, A. Donaubauer, H. Steuer, J. Jubierre, and M. Flurl, "Multi-scale geometric-semantic modeling of shield tunnels for GIS and BIM applications," *Computer-Aided Civil and Infrastructure Engineering*, vol. 30, no. 4, pp. 263–281, 2015.

[17] K. T. Park, J. Yang, and S. D. Noh, "VREDI: virtual representation for a Digital Twin application in a work-center-level asset administration shell," *Journal of Intelligent Manufacturing*, vol. 32, no. 2, pp. 501–544, 2021.

[18] A. Borrmann, J. Beetz, C. Koch, T. Liebich, and S. Muhic, "Industry Foundation Classes: a standardized data model for the vendor-neutral exchange of digital building models," *Building Information Modeling: Technology Foundations and Industry Practice*, pp. 81–126, Sept 2018.

[19] X. Lan, J. Cao, G. Lv, and L. Zhou, "Simulation method for indoor airflow based on the Industry Foundation Classes model," *Journal of Building Engineering*, vol. 39, 2021.

[20] M. Shahinmoghadam, W. Natephra, and A. Motamedi, "BIM- and IoT-based virtual reality tool for real-time thermal comfort assessment in building enclosures," *Building and Environment*, vol. 199, 2021.

[21] P. Zhao, J. Liu, X. Jing, M. Tang, S. Sheng, H. Zhou, and X. Liu, "The modeling and using strategy for the Digital Twin in process planning," *IEEE Access*, vol. 8, pp. 41229–41245, 2020.

[22] T. Ruohomaki, E. Airaksinen, P. Huuska, O. Kesaniemi, M. Martikka, and J. Suomisto, "Smart city platform enabling Digital Twin," *2018 International Conference on Intelligent Systems*, pp. 155–161, 2018.

[23] P. Zhong-Ren and Z. Chuanrong, "The roles of Geography Markup Language (GML), Scalable Vector Graphics (SVG), and Web Feature Service (WFS) specifications in the development of Internet Geographic Information Systems (GIS)," *Journal of Geographical Systems*, vol. 6, no. 2, pp. 95–116, 2004.

[24] K. M. Alam and A. El Saddik, "C2PS: a Digital Twin architecture reference model for the cloud-based cyber-physical systems," *IEEE Access*, vol. 5, pp. 2050–2062, 2017.

[25] F. Laamarti, H. F. Badawi, Y. Ding, F. Arafsha, B. Hafidh, and A. E. Saddik, "An ISO/IEEE 11073 standardized Digital Twin framework for health and well-being in smart cities," *IEEE Access*, vol. 8, pp. 105950–105961, 2020.

[26] M. Tello-Rodríguez, J. Ocharán-Hernández, J. Pérez-Arriaga, X. Limón, and A. Sanchez Garcia, "A design guide for usable web APIs," *Proceedings of the Institute for System Programming of the RAS*, vol. 33, pp. 173–188, 01 2021.

[27] A. Kychkin and A. Nikolaev, "IoT-based mine ventilation control system architecture with digital twin," *2020 International Conference on Industrial Engineering, Applications and Manufacturing*, pp. 1–5, 2020.

[28] B. Ashtari, T. Jung, B. Lindemann, N. Sahlab, N. Jazdi, W. Schloegl, and M. Weyrich, "An architecture of an intelligent Digital Twin in a cyber-physical production system," *at - Automatisierungstechnik*, vol. 67, no. 9, pp. 762–782, 2019.

[29] V. Damjanovic-Behrendt and W. Behrendt, "An open source approach to the design and implementation of Digital Twins for smart manufacturing," *International Journal of Computer Integrated Manufacturing*, vol. 32, pp. 1–19, Apr 2019.

[30] D. G. Broo and J. Schooling, "A framework for using data as an engineering tool for sustainable cyber-physical systems," *IEEE Access*, vol. 9, pp. 22876–22882, 2021.

[31] G. Steindl, M. Stagl, L. Kasper, W. Kastner, and R. Hofmann, "Generic Digital Twin architecture for industrial energy systems," *Applied Sciences*, vol. 10, pp. 2076–3417, 2020.

[32] K. Ding, F. T. S. Chan, X. Zhang, G. Zhou, and F. Zhang, "Defining a Digital Twin-based cyber-physical production system for autonomous manufacturing in smart shop floors," *International Journal of Production Research*, vol. 57, no. 20, pp. 6315–6334, 2019.

[33] E. G. Kaigom and J. Roßmann, "Value-driven robotic Digital Twins in cyber–physical applications," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 5, pp. 3609–3619, 2021.

[34] G. C. Perez and B. Korth, "Digital Twin for legal requirements in production and logistics based on the example of the storage of hazardous substances," in *IEEE International Conference on Industrial Engineering and Engineering Management*, pp. 1093–1097, 2020.

[35] C. Blume, S. Blume, S. Thiede, and C. Herrmann, "Data-driven Digital Twins for technical building services operation in factories: a cooling tower case study," *Journal of Manufacturing and Materials Processing*, vol. 4, Sept 2020.

[36] M. Schluse and J. Rossmann, "From simulation to experimentable Digital Twins: simulation-based development and operation of complex technical systems," in *2016 IEEE International Symposium on Systems Engineering*, pp. 1–6, 2016.

[37] C. L. Gargalo, S. C. de Las Heras, M. N. Jones, I. Udugama, S. S. Mansouri, U. Krühne, and K. V. Gernaey, "Towards the development of Digital Twins for the bio-manufacturing industry," *Advances in biochemical engineering/biotechnology*, vol. 176, pp. 1–34, 2021.

[38] J. Savolainen and M. Urbani, "Maintenance optimisation for a multi-unit system with Digital Twin simulation," *Journal of Intelligent Manufacturing*, pp. 1–21, 2021.

[39] Q. Qi, F. Tao, T. Hu, N. Anwer, A. Liu, Y. Wei, L. Wang, and A. Nee, "Enabling technologies and tools for Digital Twin," *Journal of Manufacturing Systems*, vol. 58, pp. 3–21, 2021.

[40] F. Tao, W. Liu, M. Zhang, Q. Qi, H. Zhang, F. Sui, T. Wang, X. Ma, L. Zhang, J. Cheng, T. Hu, H. Xu, Z. Huang, N. Yao, W. Yi, K. Zhu, X. Zhang, F. Meng, X. Jin, Z. Liu, L. He, H. Cheng, E. Zhou, Y. Li, Q. Lyu, and Y. Luo, "Five-dimension Digital Twin model and its ten applications," *Computer Integrated Manufacturing Systems*, vol. 25, no. 1, pp. 1–18, 2019.

[41] R. Darbali-Zamora, J. Johnson, A. Summers, C. B. Jones, C. Hansen, and C. Showalter, "State estimation-based distributed energy resource optimisation for distribution voltage regulation in telemetry-sparse environments using a real-time Digital Twin," *Energies*, vol. 14, no. 3, 2021.

[42] L. M. Maruping, S. L. Daniel, and M. Cataldo, "Developer centrality and the impact of value congruence and incongruence on commitment and code contribution activity in open source software communities," *MIS Quarterly*, vol. 43, no. 3, pp. 951–976, 2019.

[43] R. Helmus, T. L. ter Laak, A. P. van Wezel, P. de Voogt, and E. L. Schymanski, "patRoon: open source software platform for environmental mass spectrometry based non-target screening," *Journal of Cheminformatics*, vol. 13, no. 1, pp. 1–25, 2021.

[44] H. Khodabandehloo, B. Roy, M. Mondal, C. Roy, and K. Schneider, "A testing approach while re-engineering legacy systems: An industrial case study," in *IEEE International Conference on Software Analysis, Evolution and Re-engineering*, pp. 600–604, 2021.

[45] D. Milton P., *The ISO 9001:2015 Implementation Handbook*. Quality Press, 2016.

[46] M. C. Potter, B. Wyble, C. E. Hagmann, and E. S. McCourt, "Detecting meaning in RSVP at 13 ms per picture," *Attention, Perception, and Psychophysics*, vol. 76, no. 2, pp. 270–279, 2014.

Appendices

A Appendix 1: Case studies used

The case studies used to obtain the results shown in Section 3.3 are shown in this appendix. Table A.1 below assigns each of the figures to the respective case study.

Table A.1: Figures describing each respective case study

Name	Figures
Case study 1	Figure A.1
Case study 2	Figures 3.1, 3.2, and 3.3
Case study 3	Figures A.2, A.3, A.4, and A.5

A.1 Case study 1

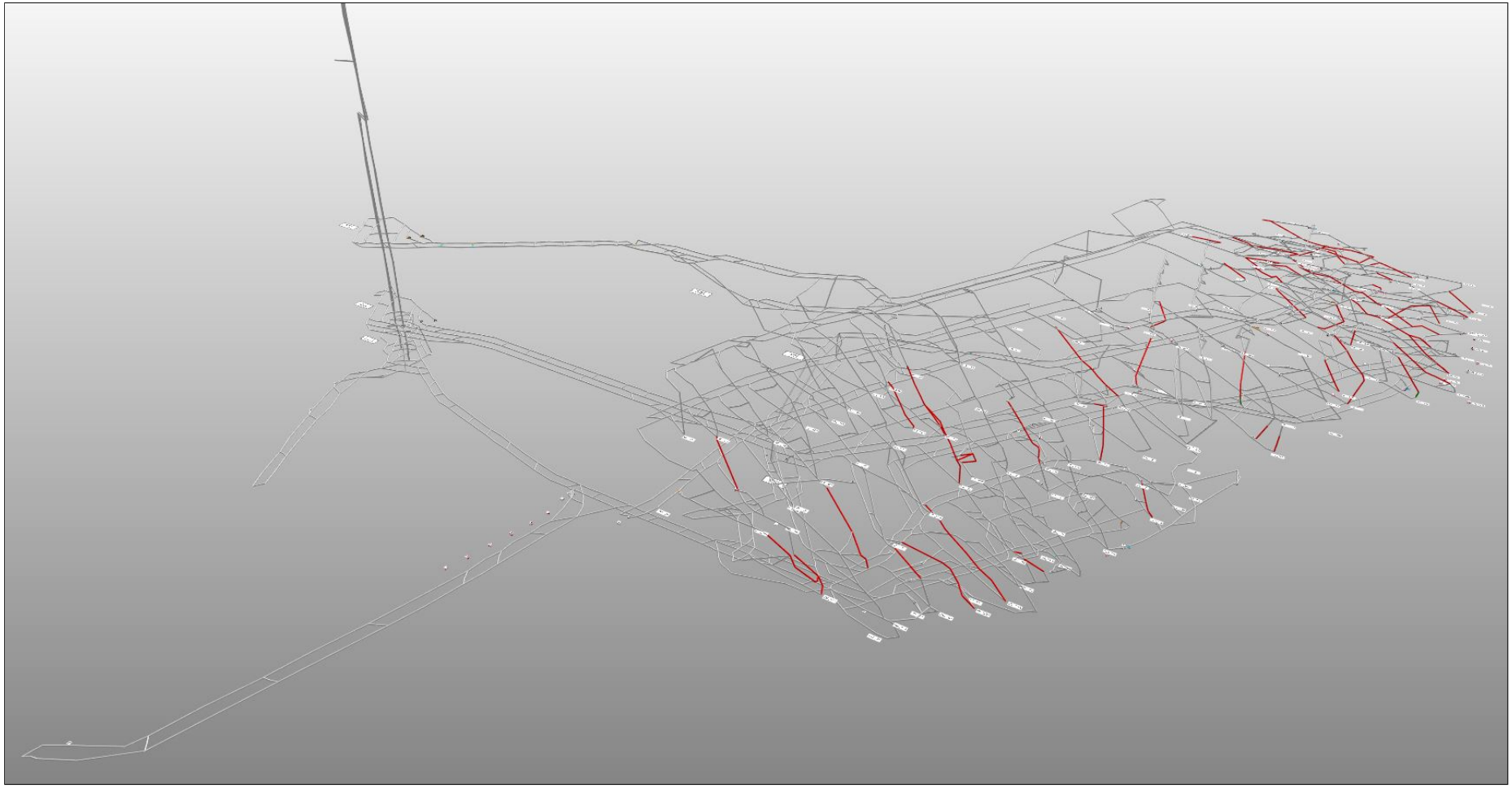


Figure A.1: Perspective view of case study 1.

A.2 Case study 3

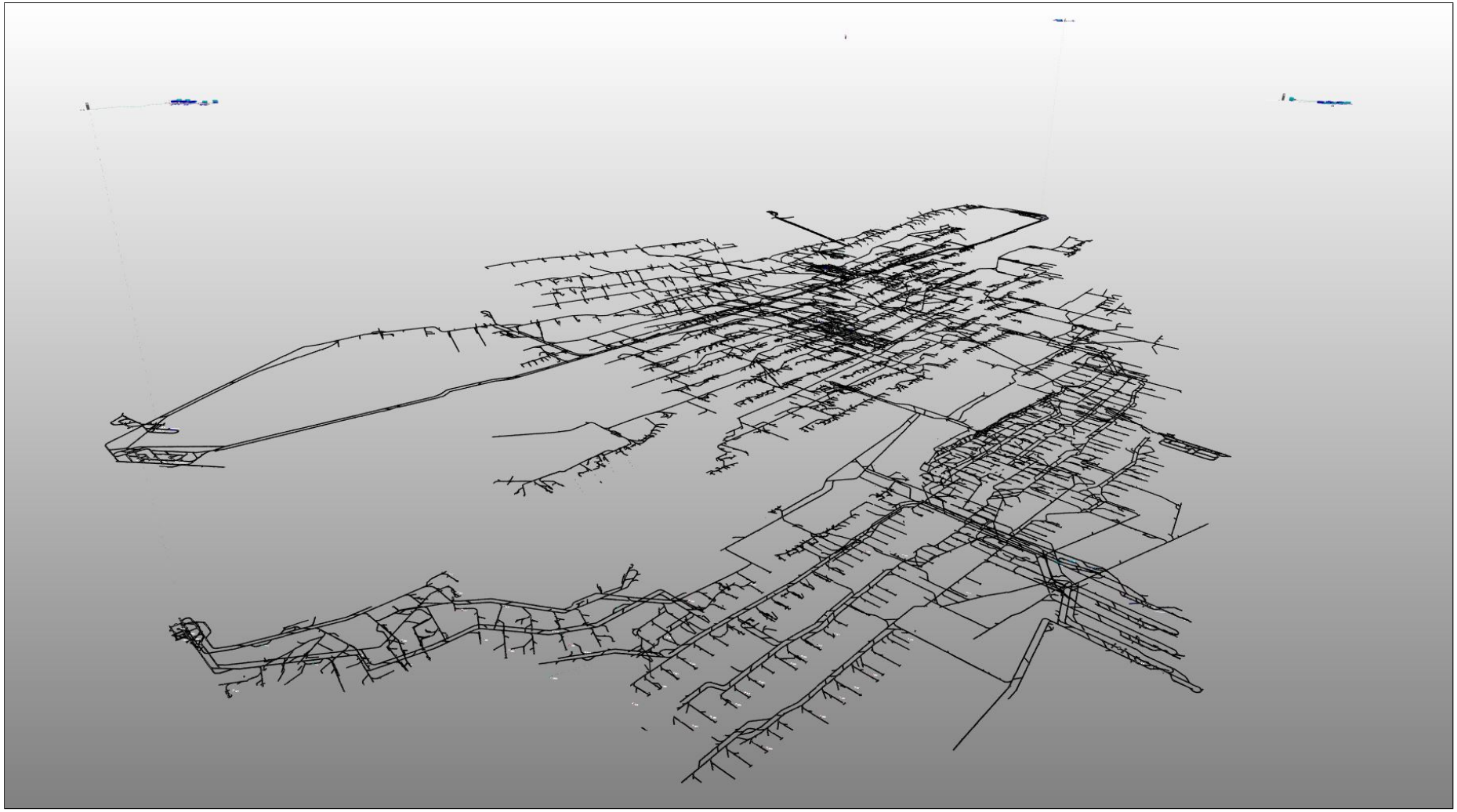


Figure A.2: Perspective view of case study 3.

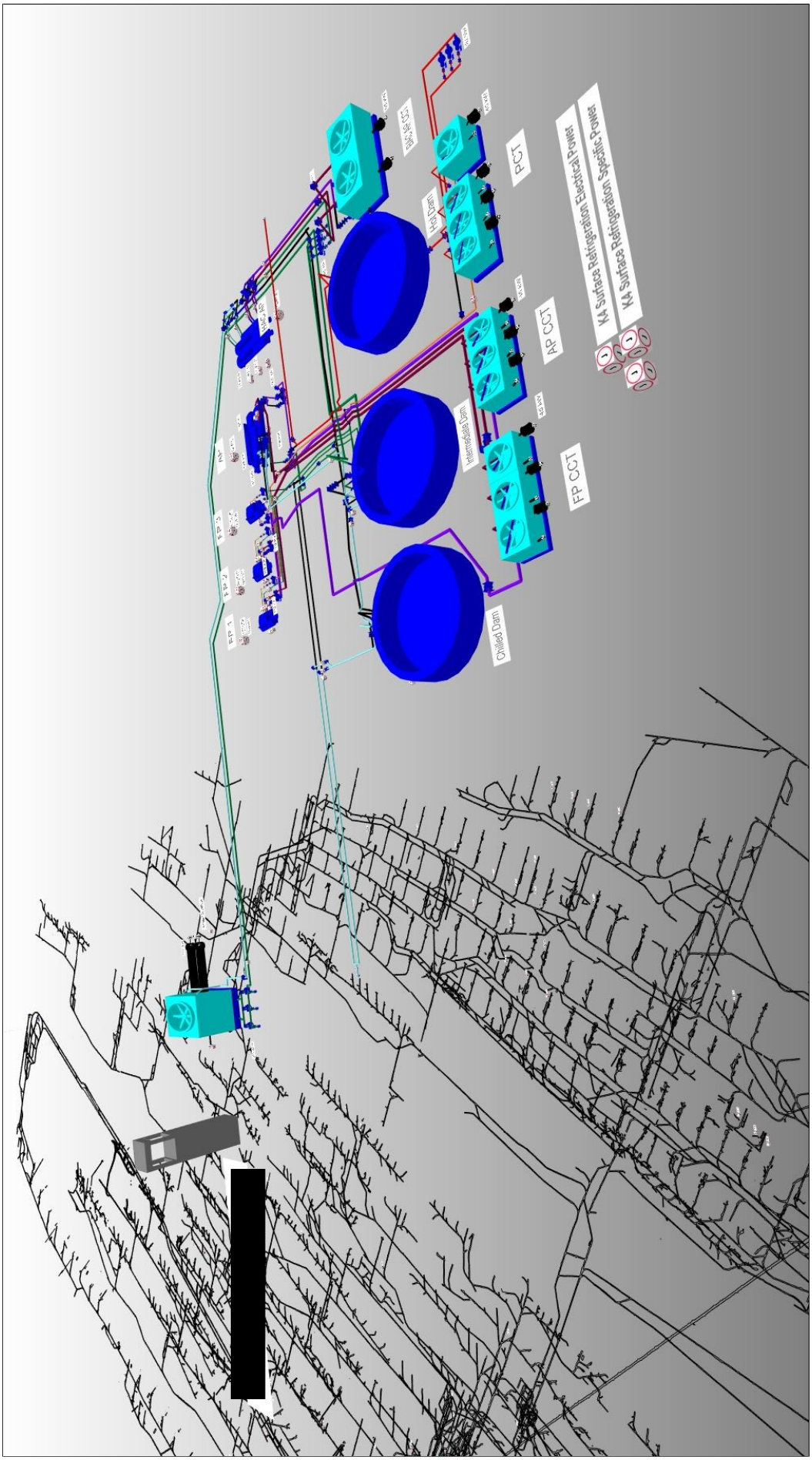


Figure A.3: Low-level view of subsystem within case study 3 with a redaction.

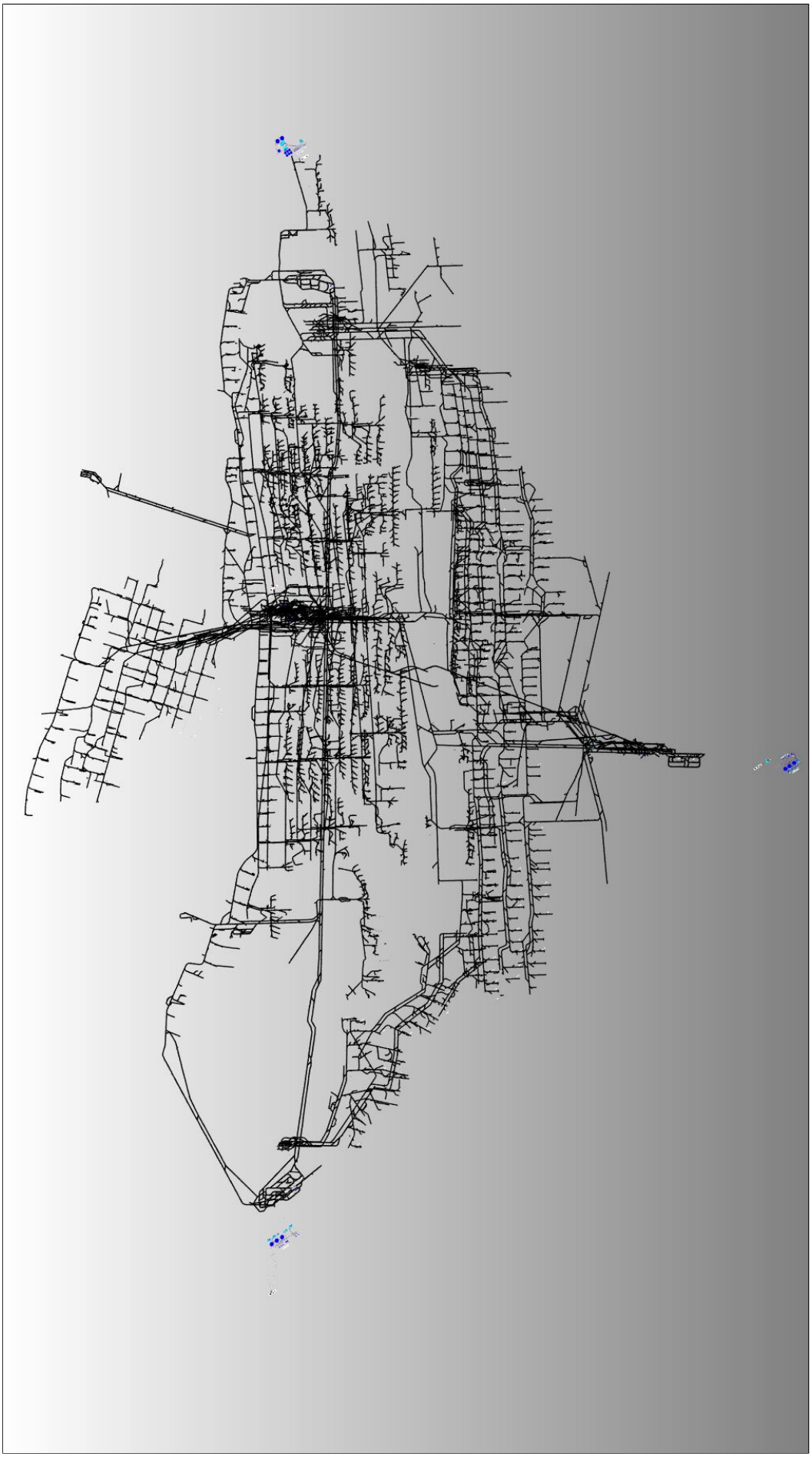


Figure A.4: Top-down view of case study 3.

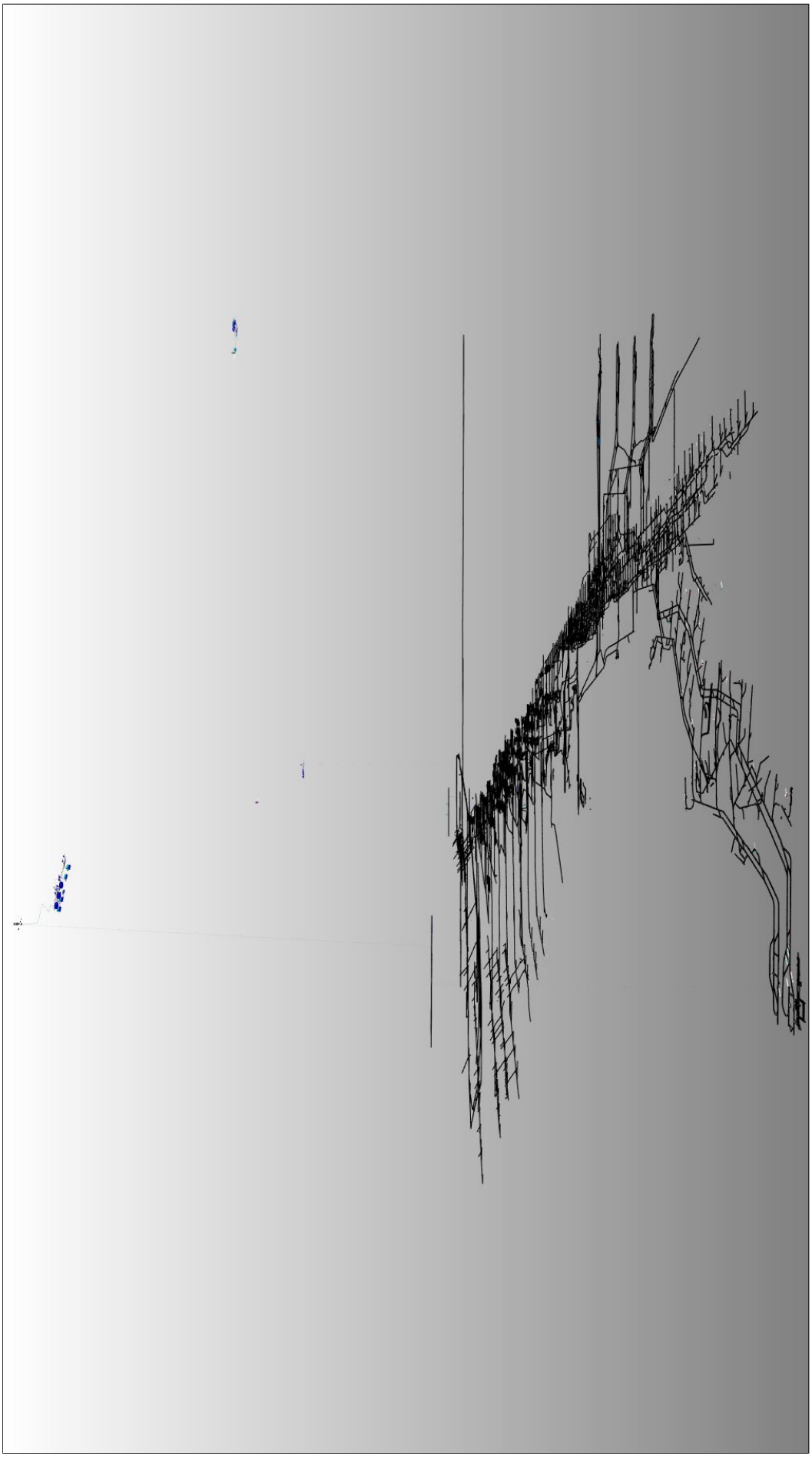


Figure A.5: Right-side view of case study 3.