TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

SEYEDSINA MIRI

DIGITAL TWIN FOR HYBRID INSTALLATIONS

Master of Science Thesis

Examiner: Professor José L. Martínez Lastra
Examiner and topic approved on 2nd of May 2018

ABSTRACT

SEYEDSINA MIRI: Digital Twin for Hybrid Installations
Tampere University of Technology
Master of Science Thesis, 51 pages, 4 Appendix pages
September 2018
Master's Degree Program in Automation Engineering
Major: Factory Automation and Industrial Informatics
Examiner: Professor José L. Matínez Lastra
Keywords: Digital Twin, Systems Engineering, Hybrid Power Module, Product Lifecycle Management, Industry 4.0

The product development and lifecycle management is constantly affected by digitalization. The same trend has been also observed in the simulation technology. The system simulation has evolved from applications with limited and specific use cases to more standardized and multi-disciplinary tools. The "Digital Twin" concept is the most recent advancement in this field where its definition is beyond a simulator. The concept arose from the "Industry 4.0" development and it can be described as a bi-directional communication between physical products data and their digital representation in the entire product lifecycle.

A hybrid power module consists of components such as an engine, a gearbox, the generator sets, the batteries, and technologies for efficiently exploiting the mechanical energy form the engine and the electrical energy from the batteries. The modular product development necessitates adoption of systems engineering approaches and principles in order to handle the product lifecycle management appropriately. Handling the product lifecycle management for the hybrid power modules encompasses the integration of disengaged elements, data, and stakeholders throughout the product development.

In order to address the abovementioned problem, model-based systems engineering approach incorporates available tools and technologies. A product lifecycle management platform and tools in hand like web services and functional mock-up interface justify the development of a digital twin application. This application must be able to reveal the adoption of system of systems view for hybrid power module development. This can be achieved by creating a reference system model and continuously enriching it with the product lifecycle data. To begin with the implementation of a digital twin application, systems engineering theories are studied, a software development lifecycle is chosen, prototypes of the application, and development technologies are selected. Lastly, the application is programmed and deployed.

The digital twin application is embedded inside a product lifecycle management platform and exploits other resources and data alongside. The application is a simplified implementation of the "V" lifecycle model in systems engineering and achieves objectives like task-centered product development, value co-creation in business processes, product data management, simulation-based, and requirements validation among others.

PREFACE

This thesis stemmed from my passion in topics of digitalization and industrial informatics, and was undertaken as a graduation requirement of the Automation Engineering program in Tampere University of Technology. The motivation, funding and the basis of this study was laid by Wärtsilä Finland Oy.

I would like to give special thanks to my supervisor Juho Könnö (Wärtsilä Finland Oy) for proposing the topic of this research, his excellent guidance and support throughout the thesis preparation. I also wish to thank Ville Kumlander (Wärtsilä Finland Oy) for his valuable contributions and assistance.

This thesis could have not been achieved without acquiring the support and insight that I received from Hybrid Platform and Systems Analysis teams (Wärtsilä Finland Oy). I also wish to thank Linnea Berg (Wärtsilä Finland Oy) for her significant guidance in thesis writing. Last but not least, I would like to thank my examiner and instructor professor José L. Martínez Lastra (Tampere University of Technology) and Wael Mohammed (Tampere University of Technology) for the academic contribution, guidance and support during my thesis writing.

Vaasa, 04.09.2018

Sina Miri

CONTENTS

LIST OF FIGURES

LIST OF PROGRAMS

LIST OF SYMBOLS AND ABBREVIATIONS

IT	Information Technology
NASA	National Aeronautics and Space Administration
PLM	Product Lifecycle management
PDM	Product Data Management
MBSE	Model Based Systems Engineering
BOM	Bill of Materials
FEA	Finite Element Analysis
SOI	System of Interest
SOS	System of Systems
BNR	Business Needs and Requirements
SNR	Stakeholder Needs and Requirements
SyRS	System Requirement Specification
ISO	International Organization for Standardization
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
ANSI	American National Standards Institute
EIA	Electronic Industries Alliance
MIL- STD	Military Standard
INCOSE	International Council on Systems Engineering
OMG	Object Management Group
UML	Unified Modeling Language
SysML	System Modeling Language
OOSEM	Object Oriented Systems Engineering Method
SDM	Semantic Data Management
IoT	Internet of Things
PTO	Power Take Off
PTI	Power Take In
SLM	Service Lifecycle Management
REST	Representational State Transfer
API	Application Programming Interface
FMI	Functional Mock-up Interface
FMU	Functional Mock-up Unit
GUI	Graphical User Interface
XML	Extensible Markup Language
DLL	Dynamic Link Library
CSV	Comma Separated Values
URL	Unified Resource Locator
SOAP	Simple Object Access Protocol
WSDL	Web Service Description Language
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
RPC	Remote Procedure Call
JAR	Java Archive
SDLC	Software Development Lifecycle
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
D3	Data Driven Documents
XHR	XML Http Request

1. INTRODUCTION

The first chapter of the thesis is dedicated to elucidating the agenda and intentions for the topic of choice and revealing the goals to be accomplished and research questions to be fulfilled. In the end of this chapter, the scope of the research and structure of the thesis is outlined.

1.1 Motivation

Product design, manufacturing and service offering have always been evolving into more efficient and better-organized approaches. In the 1960s and 1970s, within the first wave of IT, industries vastly exploited automated activities and enhanced communication in the value chain, order processing and computer aided design and manufacturing. In the 1980s and 1990s, within the second wave of IT and advent of the Internet, coordination and integration of activities and communication significantly boosted. The first two waves dramatically improved the productivity of design and manufacturing. However, industries have yet to take advantage of the third wave of Information technology, connected products data and computer sciences advancements. [1]

In the simulation technology, similar trends have been observed over the time. Simulation initially used to refer to individual applications with very limited and specific use cases. Well ahead, simulation became a standard tool to answer a specific design and an engineering needs. Afterwards, simulation-based system design was introduced to allow a systematic approach to multi-disciplinary systems, one example of which is the model-based systems engineering. The latest movement in simulation is referred to as the "Digital Twin" concept, firstly used and described by NASA: [2] "A Digital Twin is an integrated multi-physics, multiscale simulation of a vehicle or system that uses the best available physical models, sensor updates, fleet history, etc., to mirror the life of its corresponding flying twin." [3]

The "Digital Twin" in the context of product design and manufacturing is attained through integration of individual and disengaged elements throughout the product lifecycle. These elements, depending on products' nature, can be viewed as simulation models and data, operating and field data, test data, live data and product data, which are all initially disconnected. The "Digital Twin" notion originally emanates from industry 4.0 development and describes bi-directional communication and behavior of physical artefacts with their digital representation [4].

The product lifecycle management (PLM) is the essential and yet intangible procedure in product development in manufacturing firms. Therefore, adopting a functional PLM system has become an integral part of modern product development. Acquiring and implementing a well-organized and comprehensive PLM system can be a rigorous work, however; it will reimburse once it is fully adopted within the organization and activities.

Numerous benefits that stems from applying the "Digital Twin" concept as a PLM system on the one hand, and shortcomings of current system on the other hand, came to be the main motivation for the study of possible opportunities, substitutes and amendments of the existing system. Moreover, with the adoption of PLM platforms within manufacturing firms, construction and integration of new systems and technologies is meaningfully facilitated.

1.2 Justification

There is a set of objectives that justify the need for taking systems engineering approach towards product development. The "Digital Twin" in a small scale and in combination with the data available in the PLM platform aims to demonstrate several use cases that address the challenges faced in product development. These use cases amount to:

- Integration and communication of the digital twin application with an existing PLM platform and sharing the resources.
- Model-based systems engineering (MBSE) and task-centered product development.
- Verification and validation of simulation data together with requirements and test results.
- Speeding up the selection of bill of materials (BOM) based on products configuration.

Additionally, one objective is also providing a better insight into all the stages of product lifecycle with the main goal of feeding the data as inputs to business and generating values accordingly. The "Digital Twin" concept aims to provide a flawless communication among involved parties and available data in order to deliver business values. For instance, one of the outcomes of achieving the abovementioned goals is reduction in time to market [5], [6].

1.3 Research Problem

Concisely, the problem emanates from mismanagement and inability to engage relevant factors and data throughout the product lifecycle in an appropriate way. More in detail, complications can be boiled down to the following viewpoints:

- One issue can be pointed out as disconnected models and data. Models in this context are simulation models by executing of which various results are generated known as simulation data. There are also different instances of simulation models with corresponding results (also known as product configurations). So this problem can also be referred to as unobtainability of all the product configurations with their data in a common database where the results could be dynamically validated and verified with respect to the product requirements and test data.

- Moreover, product data management (PDM) and choosing the best components for the product based on the simulation results, product requirements and customer needs can be considerably enhanced as a result of a more efficient PLM system.

1.4 Research Questions

Based on the research problems stated above, the research methodology in engineering suggests observing the existing solutions, coming up with a better solution, developing it, analyzing it and finally validating the proposed solution [7].

The existing methods and practices in product development and specifically in the design phase which is the main focus in this study are not efficient. The research methodology raises questions about formalizing and structuring the product design. The main research questions to be fulfilled through this study are outlined as follows:

- How could a general-purpose system model be formed in order to serve as a reference for different product configurations?
- What infrastructure is needed for the system model to be generated and where could it be hosted?
- In what ways a system model could be augmented with other product data?
- How is the communication between a system model and the PLM data established?

1.5 Scope and Limitations

Since the research is prepared in system simulation group and is a part of hybrid power module development research team, its scope is mainly bound by the scope of activities and support from these teams within the organization. However, the results of this study can be extended to include a more comprehensive list of product development teams and activities with corresponding use cases.

In order to keep the scope of this study within time and academic structural constraints, some limitations are proposed. The application developed as “Digital Twin for Hybrid

Installations" focuses mainly on the early stages of a product lifecycle. So the main attention is given to "twinning" the product in the simulation and design phase and choice of the product components accordingly. Hence, the research questions are merely and exclusively investigated for a certain product and limited use cases.

1.6 Structure

This thesis is structured as follows:

The first chapter introduces the topic of the study, motivations, objectives justifying the need for such studies, research problems that the study aims to resolve, research questions and, lastly, the scope and limitations of study and application.

The second chapter focuses mainly on the theoretical notions and attempts to shed lights on the basics of the systems engineering through a more profound and comprehensive literature review. Moreover, a literature concerning the "Digital Twin" concept is investigated and industrial practices for similar problems are reviewed. In the end, the main conclusions of the chapter as a state of the art is presented.

The third chapter reveals a methodology of the digital twin creation. First, the research proposal for the research question of the study is presented. Then, models, tools and techniques are explored.

The fourth chapter describes the implementation of the digital twin demo application and tends to clarify the use cases employed. Furthermore, core activities for development and deployment of the application are denoted. Finally, this chapter outlines the results of the implementation.

The fifth chapter reviews accomplishments, challenges and possible future work and explores whether or not the objectives and research questions are determined.

2. LITERATURE AND INDUSTRIAL PRACTICES REVIEW

This chapter is merely dedicated to the theory, literature and industrial practices review for the topic of study. First, systems engineering principles as the backbone of the "Digital Twin" is precisely looked over. Afterwards, the "Digital Twin" and its connections with systems engineering is investigated more in detail, supported by several literature studies and previous research. And last but not least, main conclusions of the literature and industrial practices review are presented as state of the art.

2.1 Systems Engineering Review

Systems engineering originates from systems thinking that perceives the system as a whole and identifies causal relationship of variables and entities within the system. Systems engineering supports all the broad aspects and activities related to a system throughout its lifecycle from the early emergence of the need for the system by business to the definition of requirements and options, design, construction, deployment, utilization, support and, lastly, removal from service. [8]

In this section, first the system, its components and basic concepts are defined. Then the system lifecycle and principles of the systems engineering are more delved into. Next, a brief review of the systems engineering standards is presented. Afterwards, the structural and behavioral models in systems engineering are explained and, last but not least, the contribution of these model to the model-based systems engineering is investigated.

2.1.1 System

A system is comprised of a set of elements with their connections and interrelations. Once the system with the aforementioned characteristics is identified, it is bordered by systems boundary and becomes a system of interest (SOI). Elements within the system can be related to that of another system or a broader system of interest. See Figure 1. The purpose of a system is to provide a solution to a business problem. [9]

A system is not merely a product but includes coordination of personnel, activities, facilities, policies, organization, data and support that delivers an operational capability. A system can be considered as a solution to a problem and descried as a logical and physical architecture. A logical description basically focuses on what a system is and what it is aimed for. A physical description, on the other hand, emphasizes the elements of the system, how they are related and the way they should be manufactured. The logical architecture of a system as a problem domain has to be viewed as a business case without a current

logic, and it is the responsibility of the customer. The physical architecture of the system being interrelated with its logical counterpart comes later and is the responsibility of the developer or the organization that implements the system. [9]

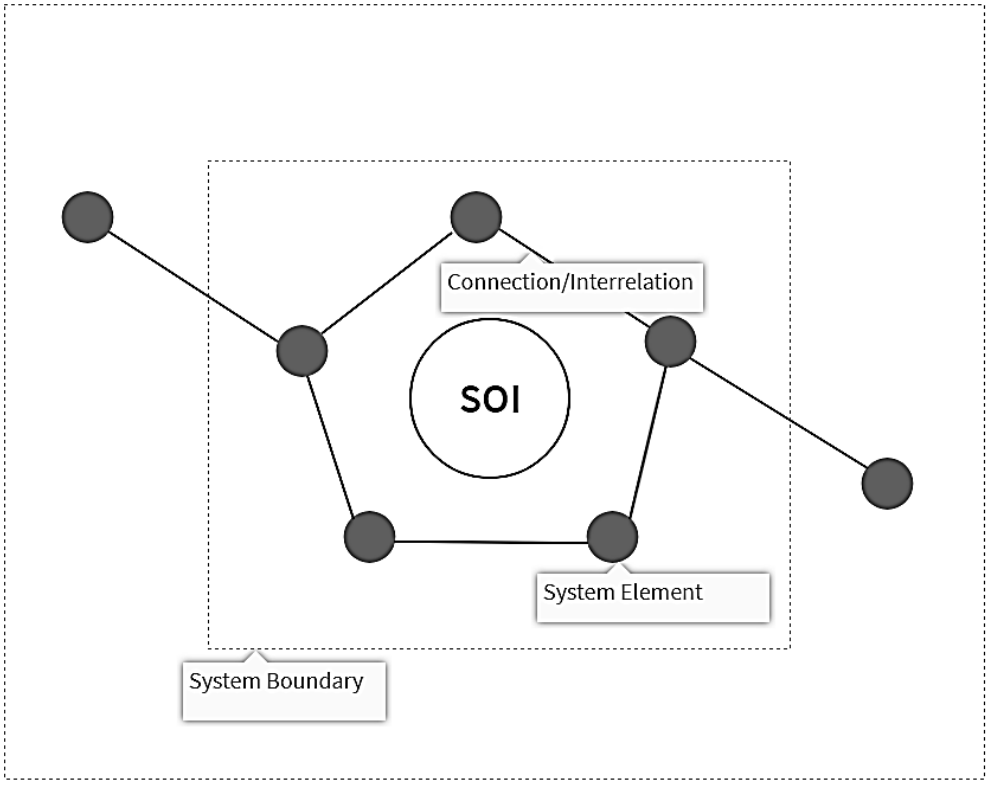

Figure 1. *System of interest: elements, connections and system boundary.*

The logical architecture of a system consists of a hierarchical structure of the system mission and its subsystems as functions that serve for the system mission. The physical architecture of a system is made up of the hierarchical structure of the system, its subsystems, assemblies and components. For instance, a hybrid vessel is considered as a system, subsystems are hybrid power module components, and assemblies are various combinations and choices of components within the hybrid power modules. It is important to note that the components in hybrid power module, such as the engine, gearbox and propeller, are each a system as they are developed independently with their own mission. A system of systems (SOS) notion comes into view when these systems are combined and tuned to operate for a broader system mission. [9]

Modularization of products necessitates implementation of systems engineering in a sense that the components of a system should cease to serve for their own purpose and mission. However, they must be optimized for their own purpose while serving for the system mission. If this view is not adopted, the system is most likely not optimized.

2.1.2 System Lifecycle

Systems have a life; they come into existence, are utilized and are finally disposed of after they have served their purpose. Throughout the system life, there is a number of activities and phases that each is built on top of proceeding activity or phase. The sum of these phases throughout the systems life is called system lifecycle. [9]

Systems lifecycle can be viewed as four main stages [9]:

- A pre-acquisition stage where the initial business need for the system is identified and the system is conceptualized. This stage is the result of business planning and includes activities to justify the need, considering the technology in use and available resources.
- An acquisition stage where the system is formalized. This stage bring the conceptualized system into existence by coordinating resources in order to comply with the business needs and requirements (BNR) identified in the pre-acquisition stage along with stakeholders' needs and requirements (SNR) and the system requirement specification (SyRS).
- A utilization stage where the system is used and evolved. At this stage the system is operated and continuously supported and maintained by the organization to modify the performance shortfalls or adopting to changes in the operation or the operating environment.
- A retirement stage where the system is disposed of. This stage is handled once the system is no longer needed or keeping and maintaining the system is not cost effective any more. Thus, the systems is disposed of and, if a substitute is needed, a new business case and planning has to be created.

During the system lifecycle stages, there is a number of parties involved also known as stakeholders. A customer is the end user of the system and, in the context of business management, a function of the pre-acquisition stage. Project management is highly engaged in the acquisition stage and supported by several disciplines, such as systems engineering, requirements engineering, quality assurance, etc. Operators are supporting and maintaining the system during its utilization stage. [9]

2.1.3 Principles of Systems Engineering

One of the key aspects of systems engineering is its top-down approach. Traditionally, engineering problems are dealt with in a bottom-up approach where components are designed and constructed and then assembled to be a part of a bigger system. However, systems engineering emphasizes on top-down approach where first, system level requirements are identified and then the system is broken down to sub-systems, assemblies and

components along with the transformation of system requirements. The Top-down approach is well defined by a "V" lifecycle model. While the system design is top-down, integration remains to be bottom-up. See Figure 2. [9]

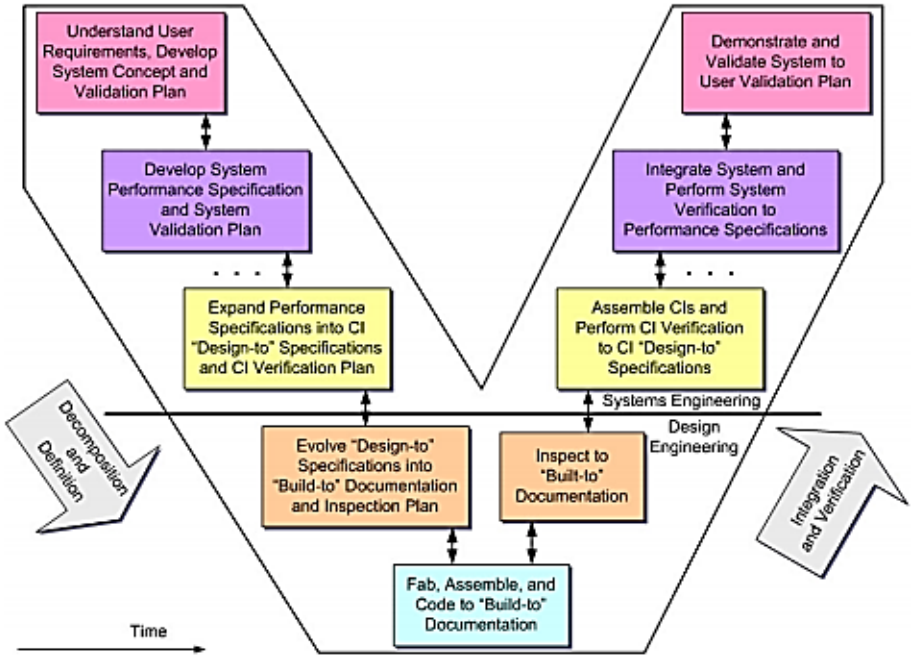

Figure 2. *"V" lifecycle development model. Design is top-down and integration is bottom-up. [10]*

Another aspect of systems engineering is requirements engineering. Initially, requirements are generated as a result of the business need for the system. Next, these requirements are translated into statements that form the basis of the logical design and, eventually, the physical design. During the transition of the requirements to lower levels, attention should be paid to translate and include all the relevant requirements. The process of handling requirements transitions is called requirements engineering. Requirements tractability is the ability of following systems design requirements in the top-down approach and inclusion of requirements in a higher level requirement in the bottom-up system integration. [9]

Another aspect of systems engineering is its lifecycle focus, meaning that all the system lifecycle stages are influenced by systems engineering. This implies that the system lifecycle should not be merely focused on the pre-acquisition and acquisition stages but also on the utilization stage where the system spends the majority of its life. [9]

Other aspects of systems engineering are optimization and balance. As mentioned earlier in section 2.1.1, fully optimized components do not guarantee am optimized system. However, designing the system by having a higher-level system mission in mind can result in an overall optimal and balanced system. This aspect is a feature of the top-down approach. [9]

Moreover, systems engineering by integrating various disciplines and management principles ensures sustainable development of complex systems.

2.1.4 Standards

Since 1930s, systems engineering has been evolving and its guidelines have been standardized in different areas [8]. In this section, the most noteworthy standards related to systems engineering, system lifecycle, and other associated factors is briefly documented.

ISO/IEC/IEEE 15288: International standard for system and software engineering with a focus on system lifecycle processes. Initially introduced in 2002.

ANSI/EIA-632: Standard of processes for engineering the systems. The Top-down approach idea is documented in this standard.

ISO/IEC/IEEE 26702: Standard for system engineering that focuses on application and management of the systems engineering processes.

MIL-STD-499B: Military standard for systems engineering.

ISO/IEC 29148: Standard for systems and software engineering with focus on system lifecycle processes and requirements engineering.

2.1.5 System Modeling

In order to model complex systems, a standard modeling tool and language are needed. For that purpose, the International Council on Systems Engineering (INCOSE) accompanied by the Object Management Group (OMG) extended the Unified Modeling Language (UML) that is a general-purpose modeling tool for software engineering. The result was the emergence of SysML that serves for modeling, design and validation of complex engineering systems. [11]

Using SysML, it is possible to model the behavior of a system and verify the system design [4]. Majority of SysML diagrams are either directly inherited from UML diagrams or modified in order to better serve for systems modeling. There is also a number of diagrams introduced in SysML that are non-existent in UML. Figure 3 shows the SysML diagrams and their relations.

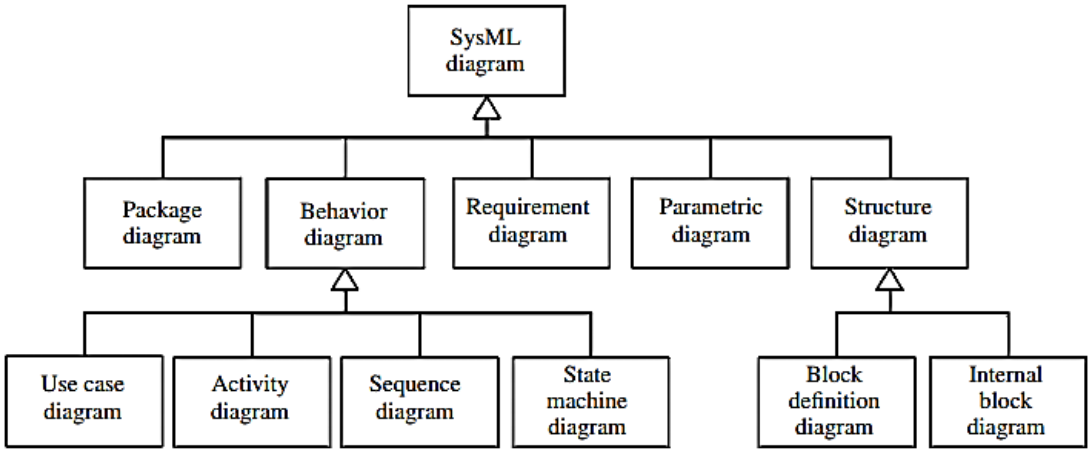

Figure 3. *SysML diagrams. [8]*

Use case diagrams allow the engineer to model systems, actors, use cases and the interactions among them. Activity diagrams depict the activities and flow of data or actions. The sequence diagram represents the message flow among objects. The state machine diagram models the state of a system upon triggering events. The block definition diagram is a substitute for the class diagram in UML and is used for modeling the structure of the system. The internal block diagram models the internal structure of individual blocks. The package diagram is used for organizing the models by structuring systems elements into packages. [12]

There are also two other diagrams that have no equivalence in UML: the requirement diagram which is used for representing systems requirements, connections among requirements and system elements. The parametric diagram is used for modeling the system parameters. [12]

As an example, the waterfall model of systems engineering “V” lifecycle, previously shown in Figure 2, can be modeled by the state machine diagram depicted in Figure 4. According to it, initially system requirements are defined and then, in the system design phase, these requirements are associated to subsystems. Next, in the system elements are integrated and, afterwards, there are system installation, evolution and decommissioning phases.

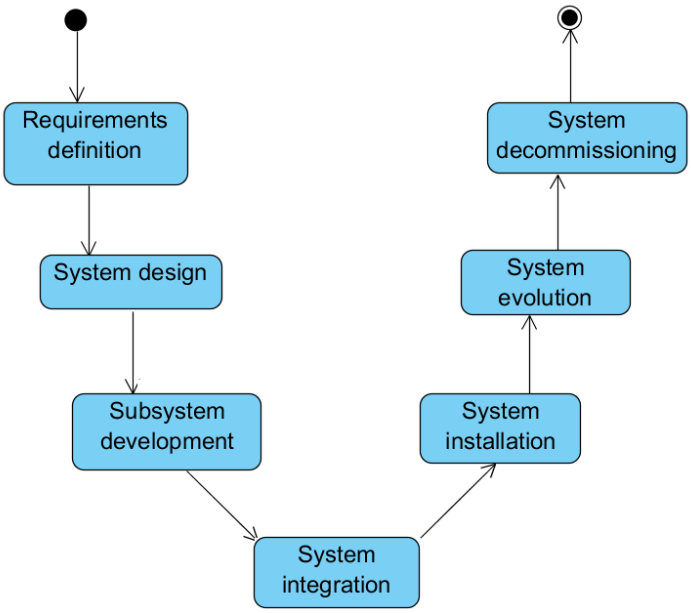

Figure 4. *State machine diagram of systems engineering lifecycle model. [12]*

2.1.6 Model-Based Systems Engineering

Model-based systems engineering (MBSE) is a methodology – a combination of models, tools and techniques that aims to support systems engineering. The INCOSE defines the MBSE as "a formalized application of modeling to support system requirements, design, analysis and verification throughout the lifecycle of the system by incorporating a set of models and simulation practices into systems engineering" [8].

Implementing the MBSE approach in product development results in numerous benefits. The MBSE improves and facilitates communication among involved stakeholders and reduces the system complexity significantly by distributing the system models to be viewed and evolved from different perspectives. Consequently, product quality is improved as a result of evolving and evaluating system models throughout the system lifecycle. Another aspect of the MBSE approach is its knowledge-driven nature and reusability that reduces model design cycle time. [8]

The MBSE approach is usually compared to traditional document-based approach where the information generated through the lifecycle of the system is stored in documents like system specifications, reports, verification plans and procedures. This hinders the maintenance and synchronization of information and reduces its quality. On the other hand, the MBSE approach by structuring the systems information and encapsulating it in system models expedites the maintenance, communication and reusability of systems requirements, architecture and design. Furthermore, digitalized manufacturing and continuous data gathering and processing enriches the models constantly with production, operating and servicing information. [6], [8]

The object oriented systems engineering method (OOSEM) is a response to the need for flexible and extendable systems design. OOSEM is an MBSE method that combines object-oriented concept with systems engineering through encapsulating the system lifecycle phases, – consistent with the "V" model, by supporting requirement specification, analysis, design, verification and validation and capturing them in objects or blocks. Therefore, it facilitates reusability, inheritance and design evolution. Moreover, OOSEM integrates MBSE with object-oriented software programming. [8]

Applying the object oriented methods to systems engineering and systems lifecycle, structures the system and generates a better insight into the systems architecture and all the underlying system elements (see Figure 5), one of the outcomes of which is tackling the missing interoperability of simulation systems with various system models. Nevertheless, a thorough approach or paradigm that incorporates simulation technology and communication through that in the whole lifecycle of the system has yet to be established.

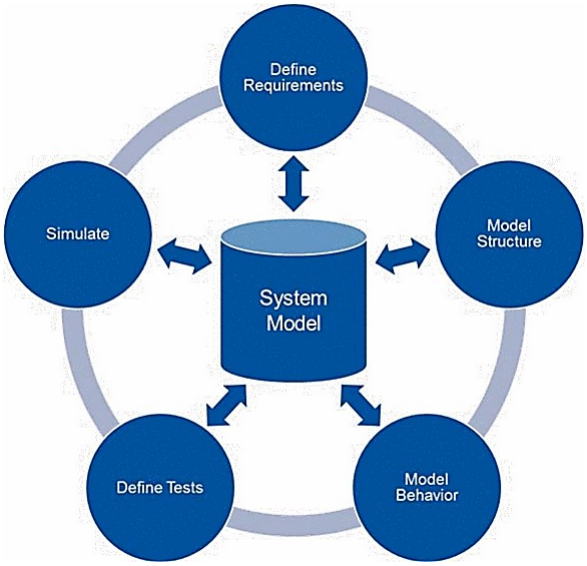

Figure 5. *MBSE methods structure systems architecture. [4]*

"Communication by simulation" is the core tenet of the MBSE [3]. MBSE by incorporating simulation technology aims to cope with the cumulative complexity of technical systems. Integration of simulation-based systems engineering with established systems engineering methods, such as the OOSEM, leads to increased cost efficiency in the development process, reduced development time, more sophisticated design and improved overall systems reliability. [4]

2.2 Digital Twin Concept Review

It is challenging to associate a precise and thorough definition with the "Digital Twin" concept as it has taken its meaning from the context and perspective where it has been developed and deployed. Efforts in implementing the "Digital Twin" concept have been

more restricted to product management and recently to shop floor and production systems [13].

Nonetheless, the “Digital Twin” in this context can be defined as a structuring element in combining the MBSE and simulation technology. The “Digital Twin” is not necessarily a comprehensive model of a physical product but a number of simulation models and other relevant data that evolve through the lifecycle of the product [14].

Up to now, some of the key features of the “Digital Twin” concept have been revealed. Since it is more of an abstract concept and its definition is inherited from its features and benefits, in this section the key features and essentials of the “Digital Twin” concept are more clarified. Moreover, building blocks for its implementation are represented. Lastly, it is investigated whether or not this paradigm is feasible and affordable.

2.2.1 Features

The “Digital Twin” concept is aimed to facilitate communication and information exchange among involved players, thus it is essential in itself to provide simplicity and accessibility while maintaining reliability. Developing the “Digital Twin” infrastructure, while keeping the MBSE method in mind ensures these goals. Likewise, the same applies to connections and communication among dispersed tools and applications, such as manufacturing, procurement, maintenance, warehousing, and filed service.

A “Digital Twin” needs to take a comprehensive approach. This means that it has to hold the systems models (system structure, components and data), and act as an analytics framework to support the system visibility and prediction. Visibility is realized in monitoring the condition and operations of the product as well as linking the disconnected pieces of information or system components. Prediction is supported by modeling techniques to foresee future behavior and state of a system. Therefore, a “Digital Twin” can act as a knowledge base for the product data and provide insight into the system.

Storing, managing and reusing the product information as mentioned above denotes that a “Digital Twin” can be viewed as a PLM system. Semantic data management (SDM) that covers both the virtual and real lifecycle of the product along with the flow of information is provided by the PLM approach. [15], [16]

In section 2.1.2, the system lifecycle was discussed briefly. The same definition may be applied to a product, assuming it to be a complex system. Thus, a “Digital Twin” as a PLM system needs to encompass the four lifecycle stages, namely introduction, growth, maturity, and decline. Applying these stages to the engineering field, as depicted in Figure 6, better clarifies the areas where a “Digital Twin” can be utilized. [17]

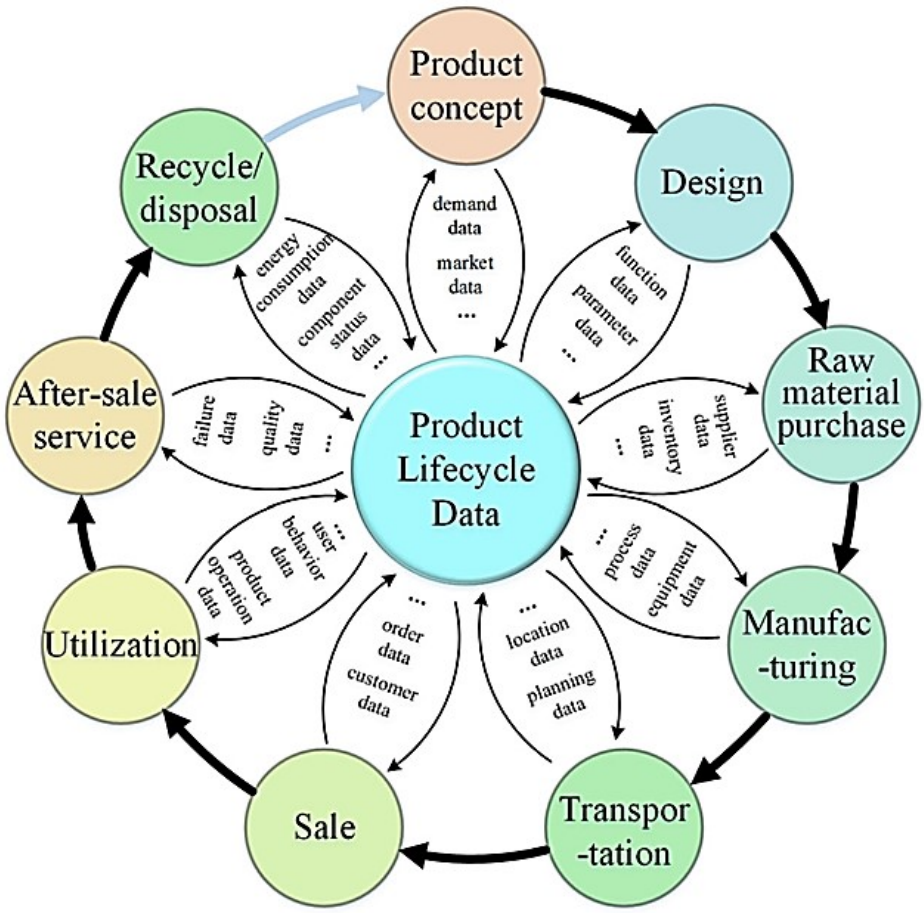

Figure 6. *Areas where a "Digital Twin" can be utilized as a PLM system. [17]*

2.2.2 Building Blocks

By combining multi-physics, multiscale, and probabilistic simulation of complex products along with models and sensory data, a "Digital Twin" replicates the life of its peer physical twin. Thus, a "Digital Twin" can be based on three building blocks: a physical product, a virtual product and connecting data that links the physical product to the virtual product. [2]

The "Digital Twin" building blocks in the design phase (conceptual design, detailed design, and virtual verification) are exemplified in Figure 7 [17]. The capabilities of a "Digital Twin" within the scope of this study are very well represented in this figure.

A "Digital Twin" needs a vast amount of data to be able to effectively virtualize the real product and predict its performance. Therefore, its true implementation requires an interdisciplinary approach. The industrial internet or industrial IoT (IIoT – industrial internet of things) takes care of sensory data retrieval. Big data analysis is required to orchestrate the vast amount of data and its integration with models.

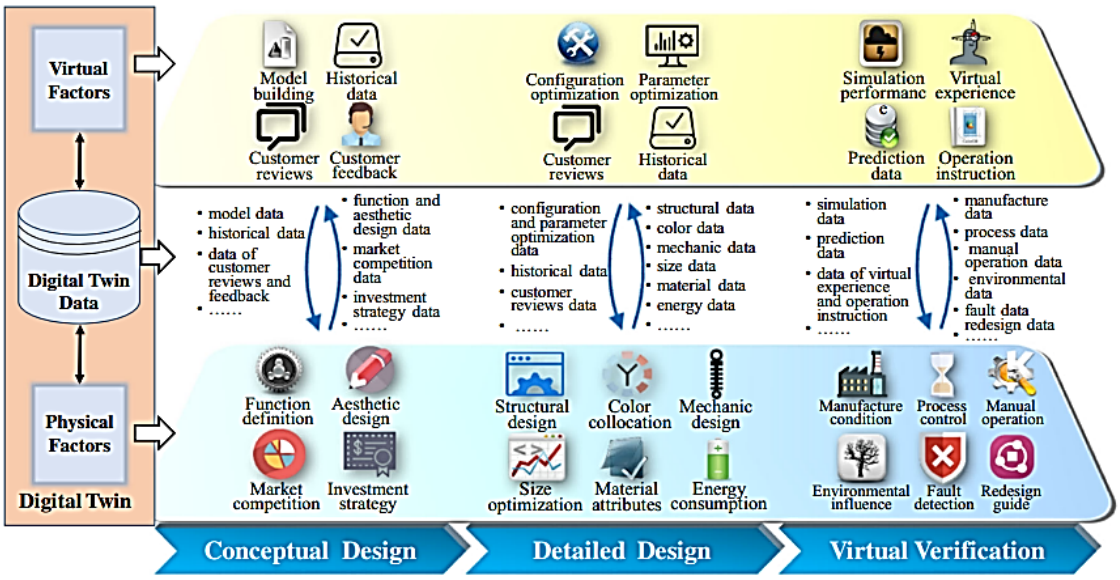

Figure 7. *"Digital Twin" in design phase of the product. [17]*

The processed data and models have to be efficiently retrieved from the isolated databases in order to interface with the end users. Web services are means of communicating among machines, through which, communication between an application and the resources in a database can be handled.

An application is another significant element through which the "Digital Twin" interfaces with the real world and communicates with involved players and other services (e.g. market pricing). The application needs to appropriately visualize the processed data and support real-time monitoring and prediction of the real product behavior.

2.2.3 Feasibility

While implementing a model-centric design and the "Digital Twin" generates numerous benefits in product development, there is a number of challenges and obstructions that have to be considered. Challenges in the way of implementing the "Digital Twin" concept are boiled down to:

- Lack of conceptual basis: one challenge in implementing the "Digital Twin" concept is the inability to apply a comprehensive model to all activities in design and production. Such to serve the vision of the "Digital Twin" is required to be extendible, interoperable and scalable. [6]
- Computational deficiencies: another obstacle in full implementation of the "Digital Twin" concept is the lack of computational capabilities. To inherit features of a "Digital Twin" requires a parallel execution of simulation and real time execution of models. Complex models can dramatically aggregate the problem and completely hinder the execution of them. [18]

- Workforce: implementing the "Digital Twin" requires a significant number of resources and workforce to produce software codes for the simulation and run-time environment. Furthermore, the massive amount of data that is constantly being produced, as a result of running models, requires unprecedented amount of data analysis work. [18]
- Cost and affordability: depending on the complexity of the models and scope of implementation, the "Digital Twin" concept can bring forth vast capital and maintaining costs. Costs are directly influenced by the abovementioned obstructions such as simulation complexity and workforce. [18]

2.3 Industrial Practices Review

This section investigates the background of the "Digital Twin" concept from the point of view of major and influential industries and manufacturing firms.

The concept of a twin was initially proposed by NASA for the Apollo program where "two identical space vehicles were built to allow mirroring the condition of the flying space vehicle during the operation. The vehicle remaining on the ground was known as the twin and, prior to operation, it was used for training and after that for mirroring the real operating conditions and real time behavior of the flying vehicle". [3]

The notion of a "Digital Twin" was originally initiated from Industry 4.0 development and denotes the virtual model development for a product, establishing a one-to-one connection and data exchange between a physical product and its virtual counterpart. Therefore, an evolving model of the product is virtually available throughout the product lifecycle. The "Digital Twin" concept can be described in three main points as suggested in [4]:

- An infrastructure that facilitates storing and accessing systems model and data.
- Clarification of systems functionality as data is processed and systems behavior is constantly monitored.
- Communication interfaces and means of correlating data and players through the lifecycle of the system.

Simulation is an integrated part of a "Digital Twin" by means of which virtual models come to real life and become experimentable [4]. Virtual models of the physical products are not only used for validation and verification, but they can also be seen as master product models with characteristics corresponding to the product [6].

Hence, the "Digital Twin" concept can also be viewed as a management system [3] where simulation technology is combined and applied to the MBSE principles. The "Digital Twin" concept highly contributes to the vision of the MBSE by encompassing system modeling and simulation in entire phases of a system lifecycle as well as for the whole

system elements. Therefore, simulation capacities can be applied to systems engineering. [4]

Once a "Digital Twin" of a physical system is placed in the center of a development process (see Figure 8), system development is reshaped dramatically [19].

Figure 8. *Digital twin of physical systems in the center of development reshapes systems development. [6]*

Before the industrial revolution, products were handcrafted by technicians as distinctive products based on a given template. However, after the industrial revolution and advent of mass production, manufacturing shifted to creating similar and interchangeable copies. [6]

There are two approaches combining the concept of customized and efficient product development. Mass customization aims to combine customized product development with near to mass production efficiency that is out of scope of this study. Another concept is referred to as the "Digital Twin" approach that stems from creating a copy of the system of interest in order to be used for establishing a relationship with real product and enabling justification for that. [6]

Major industries and companies have different views toward the concept of the "Digital Twin". Nevertheless, the core of it, the replication of a real system or product remains the same, and diverse views are a result of the different types of industries and their focus.

The goals that these industries tend to achieve through implementing the "Digital Twin" concept can illuminate its definition:

- PTC[1] (a software company) establishes a connection between a virtual and an actual product and tracks its status while it is being used by the customer. Thus, it holds a history and an overview of the product and its performance.
- Dassault Systèmes[2] (a 3D software company) focuses on complying design with product targets and the product design performance.
- SIEMENS[3] (industry, energy, healthcare and infrastructure) focuses on enhancing quality and efficiency in manufacturing.
- General Electric[4] (aviation, healthcare and power sectors) attends to forecast condition and performance of their products.
- TESLA[5] (an electric vehicles manufacturer) focuses on creating a "Digital Twin" for each manufactured car, therefore allowing real-time condition monitoring of the cars.
- Deloitte[6]: focuses on manufacturing processes and delivers a digital twin as a service in order to generate business values [20].

2.4 State of The Art

In this section, a comparison of product development in the form of "as-is" vs. "to-be" is presented. This assessment is made from the technology viewpoint and based on experience, observations and communication in the core team[7] meetings. The comparison is simplified and is limited to a few general use cases in the product development lifecycle.

Once the business case for a product is created, a research group for design, development and delivery of the product is formed. A research group in the research and development organization is structured in a way to get all the competencies and stake holders together in order to ensure achievement of the research project goals. The roles in the team are defined to support the final product functionality, mechanical design, engine operation, performance and control, system simulation and system integration. [21]

Once the research team is established and members know their responsibilities, the process of project requirements definition and value proposition starts.

Currently, the project requirement documentation is Excel-based and collected by the members of the core team. These requirements account for the detailed specifications of

[1] https://www.ptc.com
[2] https://www.3ds.com
[3] https://www.siemens.com
[4] https://www.ge.com
[5] https://www.tesla.com
[6] https://www.deloitte.com
[7] Hybrid Platform Team, Wärtsilä Finland Oy

all the components in the module, and they are vessel-specific. This process can be quite tedious and time-consuming due to possible miscommunication within the organization and unavailability of data. Also, the independent function of separate teams for different system components is one reason behind this.

A more efficient way of handling project requirements is systems engineering approach as suggested in section 2.1.3. This means that the vessel requirements could be initially defined by the customer. Then in the next step, requirements from the customer's point of view would be translated into general technical requirements and consequently broken down and handed over to appropriate teams within the organization in order to collect the detailed specifications.

Currently, the systems simulation engineer who holds the responsibility of integrating the systems components and simulating the system behavior as a whole, receives the requirement specifications and tunes the simulation models on the basis of them. Lack of information about system components and vessel requirements due to miscommunication within the organization hinders and slows down simulation engineers' work.

The process of systems simulation could be significantly enhanced through the use of model-based systems engineering methods. Different configurations of each component could be generated and a system model created on the basis of the requirements. System model creation can be significantly boosted by exchanging component configurations and simulating accordingly.

Traditionally, once the simulation results are generated and the system design agreed upon, the production and integration of the product components starts. The process of production and integration is handled in different production units. Neither at this stage, nor the previous ones, the customer is necessarily fully aware of the product status until the final product is ready for delivery. Moreover, development team may not be able to receive timely feedback about production and system status.

Production and integration of system components could be boosted through the use of the Internet of Things (IoT) and continuous updates of product status throughout the development lifecycle. Moreover, all the players in the system development could receive a real-time status and report of the system.

The "Digital Twin" vision is not merely restricted to product development but aims to support its physical counterpart in all other lifecycle phases, such as operation until the product is disposed of. Thus, after product development, the use cases and focus are mostly on monitoring the system condition and delivering services, such as predictive and preventive maintenance.

3. RESEARCH PROPOSAL AND METHODOLOGY

This chapter brings the methodology of the "Digital Twin" concept to light. First, the proposal for the digital twin interface is illustrated by describing the hybrid power modules, their digital counterpart and corresponding sources of data, and enumerating a number of scenarios and use cases to be implemented in the solution. Next, the models used and generated to support the development of this study are elucidated. Finally, Tools that have been used throughout the study and development of the "Digital Twin" demo application are presented.

3.1 Proposal

In order to tackle with the research problems of this study, the idea of digital twin interface is proposed. The digital twin interface is intended as a web based application that not only maintains a connection to other web resources such as PLM data, but also serves as a reference system model that continuously evolves along with the physical product lifecycle. Therefore, a replicate of a physical product is accessible on the web, enriched with all the physical product data that spans from design phase data to the operation and maintenance data. Besides, the system model generated for one product, can be utilized as a reference model for other similar products in order to streamline the product development and optimize time and resources within the organization.

In the scope of this study, a "Digital Twin" is researched as thoroughly as possible. However, its implementation is narrowed down to a manageable number of scenarios and use cases since its full implementation may not be simply achieved due to time constraints and the available resources. In this section, a description of a product (hybrid power module) is given and the concept of a digitalized product, data sources and information retrieval is discussed. The scenarios and uses cases that will be put into practice are reviewed. Within this scope, the "Digital Twin" concept is implemented to verify that the product requirement specification is consistent with the product intents and customers' expectations [6].

3.1.1 Product Description

Research, study and implementation of the "Digital Twin" concept and applying the systems engineering principles are targeted for hybrid power modules (Figure 9). Each hybrid power module is made up of set of components that is tuned and harmonized to serve for a specific vessel type. [22]

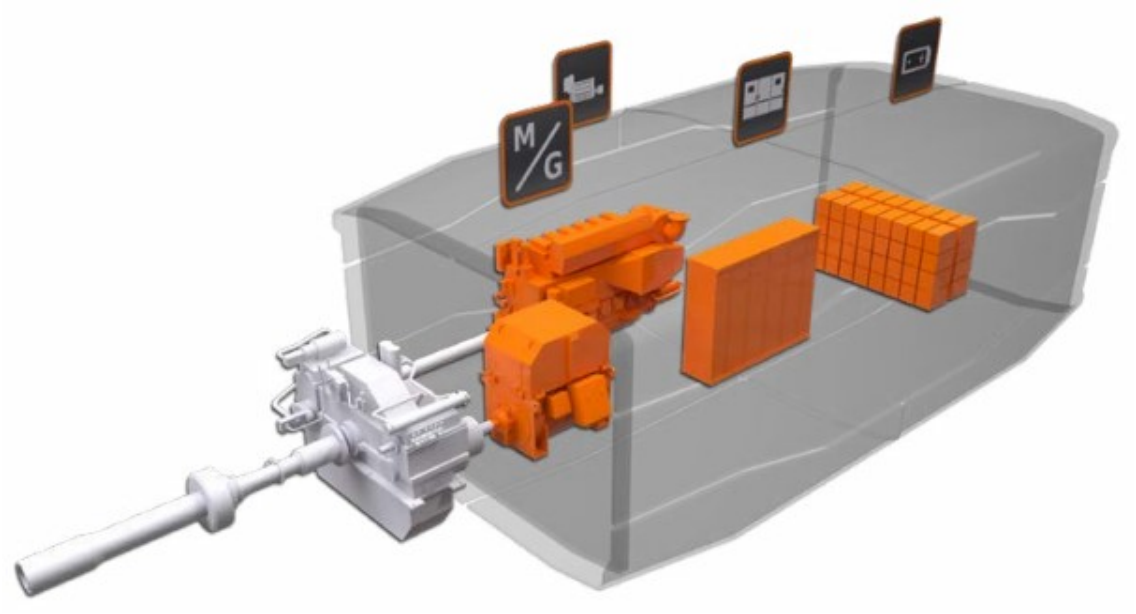

Figure 9. *Integrated hybrid power module. [22]*

A hybrid power module is an integrated set of components that work as a system. It includes a main engine with a clutch, a power take off / power take in (PTO/PTI) that harnesses the mechanical energy of the drive shaft and converts it to electrical energy, a two-speed gearbox, an energy storage system, a DC link and power drives, and an energy management system. [22]

The hybrid power modules aim to combine customer needs with the organization's expertise and experience in the marine industry to deliver the state of the art power module solution to continue being competitive in the marine market while maintaining marine regulations. Some values that are aimed for by delivering hybrid power modules boil down to smokeless start of engine while the vessel is in the harbor, emission control, damping the load fluctuations, and cost efficiency.

The hybrid power modules were nominated for study and application of the "Digital Twin" concept and systems engineering due to various reasons. From a lifecycle management point of view, upon starting this study, development of product was in the pre-acquisition phase. Therefore, there has been room for many research and development studies and discussions. Another reason is the modular nature and customizability of this product for different operating profiles, well justifying the need for systems engineering approaches.

3.1.2 Data Types and Retrieval

A digitalized product stems from the fact that product data is constantly being collected throughout the product lifecycle phases. This data could span from a very basic requirements definition in the pre-acquisition phase to design requirements and data (simulation data and results), production and manufacturing, operating and servicing.

It is assumed that the product is digitalized and all the product data is available in digital format. However, there is a lack of a management system where the actors can query their needed data, observe the system performance and predict the system behavior [3].

Digitalized and smart products generally hold three types of data: architectural data, component data, and operating or usage data [15].

- Architectural data is generated mainly in the pre-acquisition and acquisition phases and usually stored in local and dispersed databases. The requirements definition for a product is an example of architectural data. Detailed product requirements are generated in several sessions of meetings and, in some cases through the use of history data. Such data forms the basis and initial conceptualization of a product.
- Component data is generated during the design and analysis phase and stored in a cluster or local machines. Simulation data generated by successfully running the simulations is an example of such data. Furthermore, through the use of data analytics, such data can be analyzed to produce further knowledge in the system components domain. Product data and system components are available as services.
- Test and operating data is also available in specified databases and restricted for authorized users. Such data is attained through testing the product before release and then by monitoring the performance while the product is in operation. Similarly to component data, a significant amount of data analysis is required for the operating data to discover the hidden knowledge and useful patterns.

One of the goals of the “Digital Twin” concept is the integration of heterogeneous data and information from different sources. These sources of data are not in a unified structure, and that makes the process of data retrieval challenging.

As noted previously, data sources are quite dispersed and disconnected. The combination of a PLM[8] and an SLM[9] (service lifecycle management) platforms, resolves the problem of data access and integration to some extent. Through the use of REST (Representational state transfer protocol) API, it is possible to fetch some product requirements, simulation models, lifecycle data, and results from the PLM and SLM platforms.

[8] 3DEXPERIENCE platform
[9] Teamcenter platform

Retrieving and integrating data from all the different sources is out of scope of the "Digital Twin" demo application. Thus, the main focus is on retrieving data from the PLM platform that holds the components data and is used for product data management.

3.1.3 Scenarios and Use Cases

As the title of this thesis suggests, the digital twin interface is to be implemented for the hybrid vessels. Thus, the main focus is on the hybrid power modules as they hold the features of complex systems (hybrid power modules explained in section 3.1.1). Nevertheless, it could be extended to include other modules or products as well.

The set of roles and scenarios considered for the implementation of the digital twin interface:

- Customer as one of the actors logs into the digital twin interface and:
  - Views the status of the product and its development phase.
  - Defines the requirements for the product.
- Product Manager as another actor logs into the digital twin interface and:
  - Views the customer requirements and translates them into technical and systems requirements to be used by simulation engineers.
  - Views the reports generated by the simulation engineer on the basis of chosen simulation configurations and approves the appropriate report.
  - Handles the product data management based on approved simulation configuration.
  - Checks other relevant information regarding the vessel and their status.
- Simulation Engineer logs into the digital twin interface and:
  - Selects one of the generated requirement specifications for the hybrid power module.
  - Views the systems requirements (functional and non-functional requirements) and appends documents or simulation configurations to the test cases based on the requirement specifications.
  - Selects different simulation configurations and checks the systems components.
  - Based on simulation results, requirements and test data, handles the systems analysis and generates a report for the selected simulation configuration.

Based on the description given for the scenarios, it can be implied that actors involved in the system interact and form a closed loop. Furthermore, a use case diagram (Figure 10) is created to better envisage the system, actors, use cases and scenarios. This use case diagram is the basis for the development of the application.

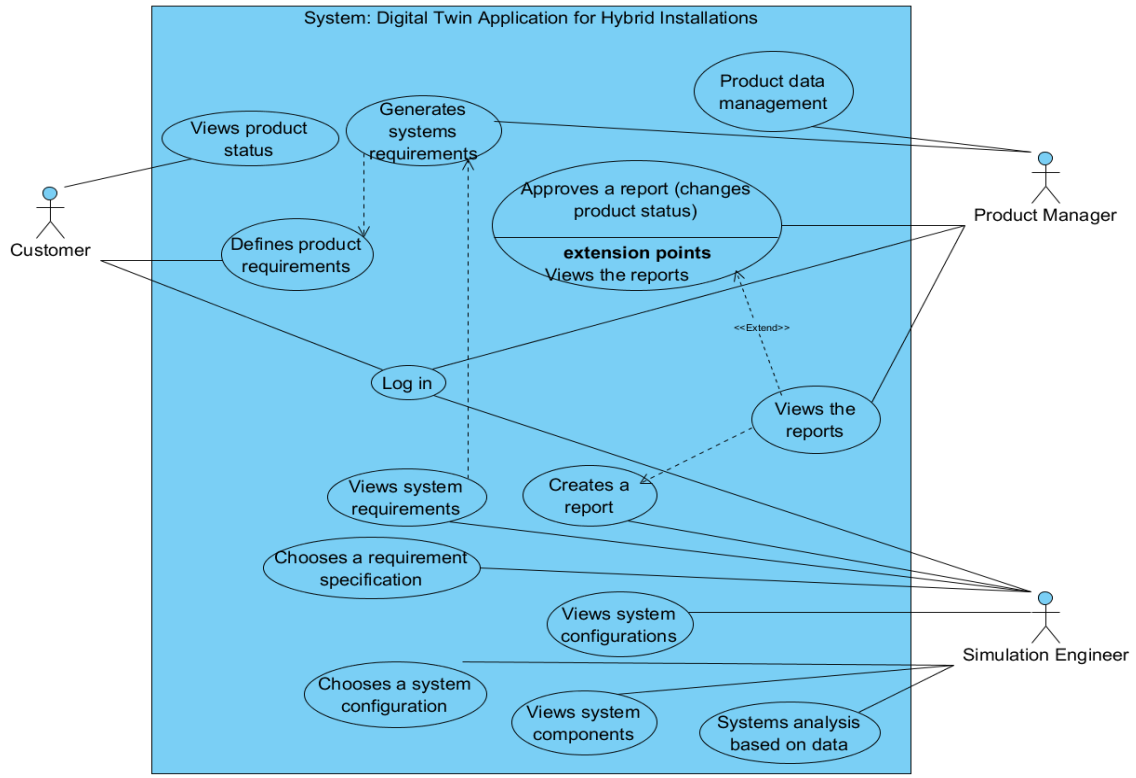

Figure 10. *Use case diagram for the digital twin interface.*

In order to further elaborate the scenarios described for the application, sequence diagrams are created based on the use case diagram in Figure 10. A sequence diagram, while maintaining simplicity, shows all the interactions between each actor and the system.

Figure 11 shows the sequence diagram for customer interactions with the digital twin interface. The customer visits the digital twin application and logs into the system with relevant credentials. Upon a successful log-in, product status can be accessed. Moreover, the customer defines the requirements where they are added to the list of product requirements.

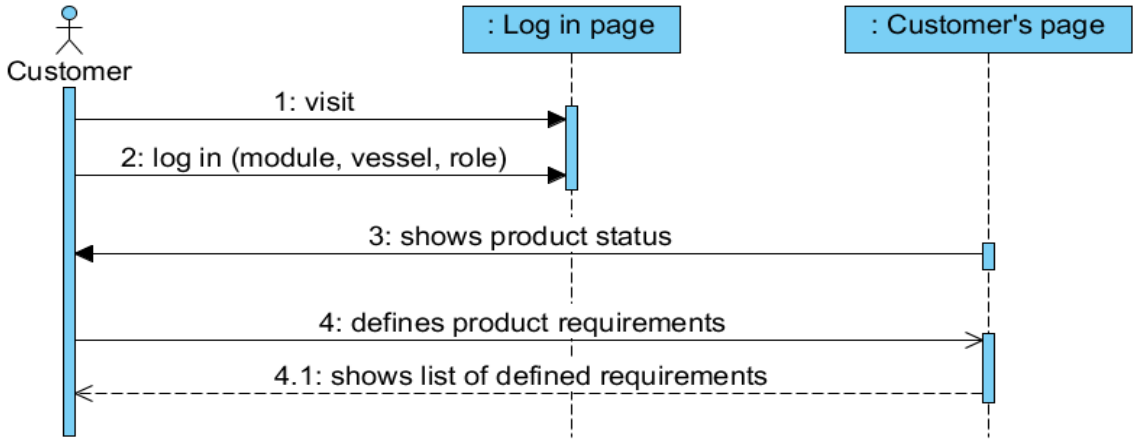

Figure 11. *Sequence diagram showing customer interactions with the system.*

Figure 12 shows the sequence diagram for the simulation engineer interactions with the digital twin interface. The simulation engineer visits the digital twin application and logs into the system with relevant credentials. Once logged-in, the simulation engineer chooses a requirement specification for the given product and the digital twin interface shows the system requirement specification. Next, the simulation engineer chooses a simulation configuration based on the system requirements and verifies the system components. Finally, the digital twin interface shows the simulation, test and requirements data using data tags where they can be used for analysis and generating reports.

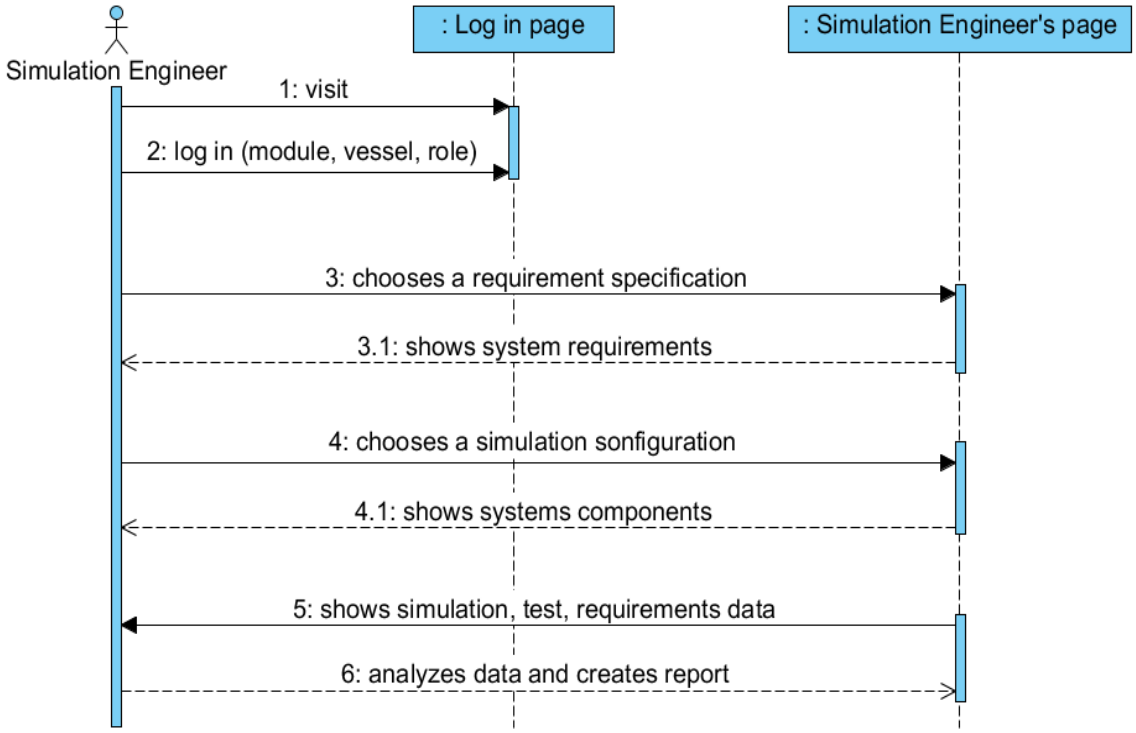

Figure 12. *Sequence diagram showing simulation engineer interactions with the system.*

Figure 13 represent the product manager interactions with the digital twin interface. The product manager visits the digital twin application and logs into the system with relevant credentials. Once logged-in, the digital twin interface shows the requirements defined by the customer. Then, the product manager generates the technical requirements specification for the product. Moreover, the reports previously generated by the simulation engineer are shown to the product manager where a simulation configuration can be approved. Finally, based on the approved simulation configuration, the product manager selects the bill of materials for each component through the established connection with PDM.

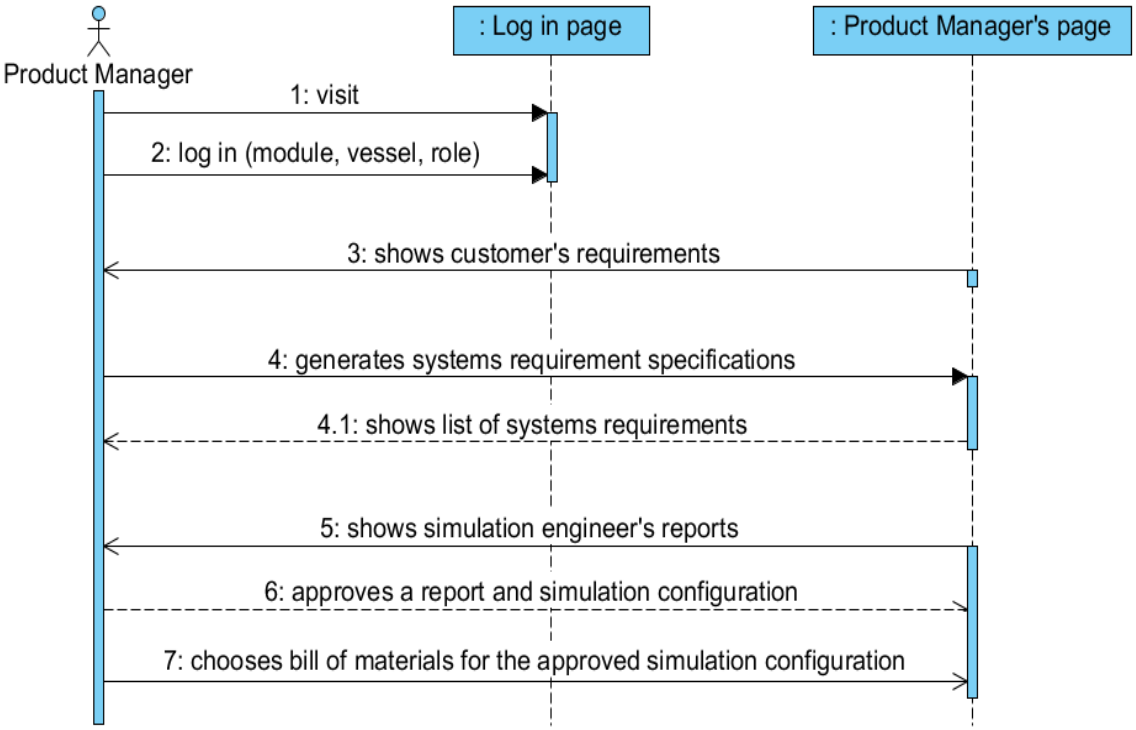

Figure 13. *Sequence diagram showing product manager's interactions with the system.*

3.2 Models

This section is about the models used and generated in this study. The simulation models are developed for any specific vessel type by a system simulation engineer. Using various simulation models from different vendors is not quite efficient for the "Digital Twin" development due to inoperability and license issues. Therefore, by using the functional mock-up interface (FMI), any specific simulation model is converted to functional mock-up units (FMUs) to facilitate interoperability and integration of heterogeneous models.

3.2.1 System Simulation

This section, presents a brief description and overview of simulation technology as a key element in the "Digital Twin" concept; in a manufacturing framework.

Having its basis on the theories of probability, simulation in many areas of technology has proven to be a trusted and affordable approach to predict and observe the behavior of systems and real life phenomena. Simulation in the manufacturing perspective is computer-aided design and analysis of manufactured products in order to predict and improve the systems design and behavior by manipulating and modifying the involved and controlling parameters. [23]

Following an established logic, simulation models are created with a certain degree of precision and simplified assumptions to replicate the real life systems and processes. Simulation data generated by successfully running simulation models open the door to numerous applications and possibilities. Once the data is stored, many scenarios of statistical analysis and validation can be applied [23].

There is a variety of approaches to facilitate simulation of different real life phenomena, systems and processes. The key element in them is the applicability of results to the real system with a sufficient accuracy. The most well-known approaches to simulation technologies can be listed as: [4]

- Block-oriented simulation (e.g. Simulink[10], GT-Suite[11])
- Declarative modeling (Modelica[12])
- Discrete-event simulation
- Finite element analysis (FEA) simulation
- Mechatronics systems simulation

System simulation is the essence and main source of component data. Different kind of simulation approaches can be incorporated to achieve the replication of systems behaviour, depending on the systems nature. The choice of the simulation approach and, subsequently, the framework in which the simulation model has to be developed is affected by the available resources and capabilities within the company. Among different approaches to and frameworks for system simulation, the block-oriented approach is chosen for the simulation model development of fishing vessel that holds the hybrid power module.

Simulink is a tool for modelling and analysing dynamic systems through accessing MATLAB data files and functions. With a graphical user interface (GUI), it exploits the block-oriented simulation approach [24]. Simulation models are either designed or chosen from Simulink libraries and toolboxes. The main challenge in creating the simulation model is converting the physical system to a set of equations that forms the Simulink building blocks. [24]

Every Simulink block consists of one or more inputs and outputs. Therefore, a number of blocks are connected together to form a greater entity. Once the simulation model by a meaningful connection of blocks is generated, simulation may run. By running the simulation, the model is converted to a set of equations and solved by the hosting simulation solver. Then the simulation results are generated and may be used for different purposes such as visualization and data analysis. [24]

[10] https://www.mathworks.com/products/simulink.html
[11] https://www.gtisoft.com
[12] https://www.modelica.org

A simulation model of each system component is created similarly to form a sub-model set to cover the system in whole. Therefore, connecting the sub-models generates the model of the whole vessel. A simulation model of the hybrid power module for a specific fishing vessel was created through the connection of sub-models representing the module components. Figure 14 shows the Simulink model of the engine that is a complex sub-model of the hybrid power module.

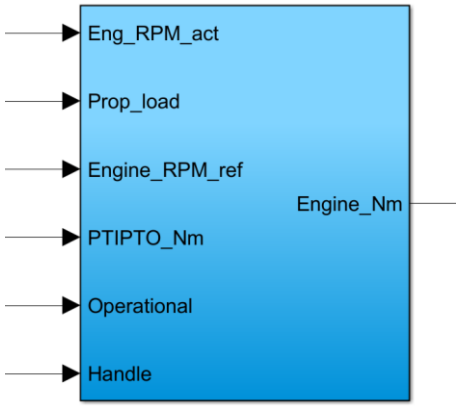

Figure 14. *Simulink model of the engine. Inputs are each connected to blocks inside the engine model.*

3.2.2 Functional Mock-up Interface (FMI)

One of the challenges in implementing model-based system design for development of complex systems is heterogeneity of systems models. A number of tools and methods have been introduced to accommodate to such issues, each with its strengths and deficiencies. The functional mock-up interface (FMI) is a recent response to the need for a standardized and tool-independent framework with the goal of co-simulation and model exchange. [25]

FMI development was initially started by Daimler AG with the aim of exchange of simulation models among vendors and various simulation environments. Now this standard is being supported and maintained by Modelica Association[13]. This standard encompasses conversion of simulation models to functional mock-up units (FMUs). [26]

The exported simulation models (FMUs) act as black boxes that serve for port-based communication by encapsulating concepts for simulation algorithm interaction [4]. Each FMU contains a model description xml file, source code and libraries. The xml file holds the description of the simulation model, inputs, outputs, parameters and their values (See Figure 15). The source code and libraries are in the form of a dynamic link library (dll)

[13] https://www.modelica.org

in order to ensure the protection of source code which holds the functionality of FMU. [27]

The mechanisms through which FMUs interact with the host simulator is provided by:

- Model Exchange (FMI-ME) where the FMU does not hold its solver and requires the host simulator to perform numerical integrations [25].
- Co-Simulation (FMI-CS) where each FMU holds its own solver or execution mechanism. Use of FMI for co-simulation is aimed for interoperability of different simulation models [25].

As it was mentioned in section 2.1.5, SysML is used for modeling complex systems. SysML supports multi-modeling and co-simulation and therefore it is fully compliant with FMI [28]. In other words, each FMU demonstrates the behavior of one or more SysML blocks [4].

The simulation model of a fishing vessel elucidated in section 3.2.1 can then be exported by a specific FMU exporting tool to an FMU. Exportation can be either performed for individual simulation models or for the whole model as a single FMU. The exported FMUs resemble the simulation models with the difference that internal blocks and functionality are hidden. Hence, for tuning and configuring the FMUs, it is only possible to modify numerical values of parameters that are used inside the FMU. This can be done either by tuning the model variable values in the model description XML inside the FMU package (Program 1 in Appendix A) or by importing the FMU into a simulation environment and changing the parameter values (Figure 15).

Once the simulation models are successfully exported to FMUs, files with the .fmu extension are generated. FMUs are each a package containing a model description XML (an example of which is partly shown in Program 1 in Appendix A), C code and supporting libraries. These FMUs can then be imported to supporting modeling and simulation tools. These FMUs in the simulation environments can be coupled with other FMUs or models. In Figure 15, the corresponding FMU of a simulation model of the engine (Figure 14) is generated and then imported to a simulator. In Figure 15 the FMU of the engine is selected. With all the simulation details being hidden, it is only possible to modify the numeric values of the parameters.

Once the model composed of FMUs is created, it is possible to configure the component parameter values or to add a different abstraction of the model for each FMU. Execution of the process runs the simulation and stores the results in the csv format into the database.

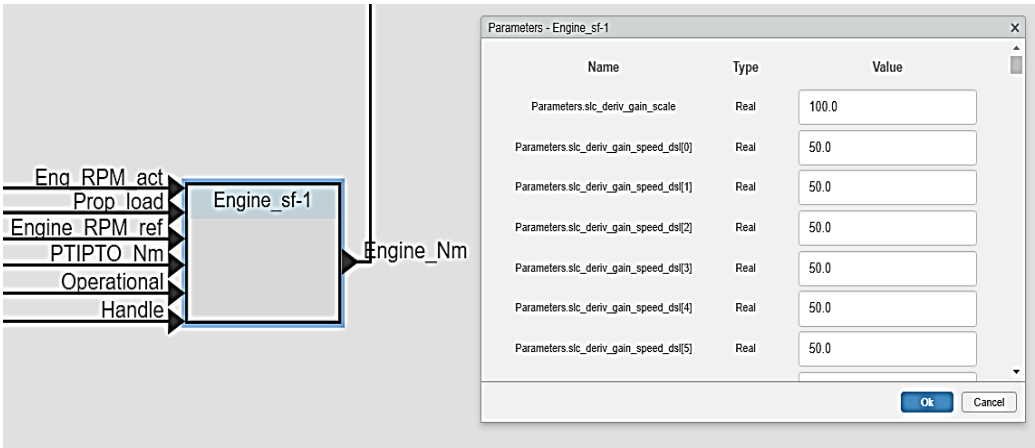

Figure 15. *FMU of engine in a simulator. Simulation details are hidden.*

Since the FMI is an open standard and tool-independent, it can be utilized in any framework, making it an appropriate candidate for use together with the digital twin interface. Because a "Digital Twin" tends to deliver an overall picture of the system, the simulation details have to be hidden from the systems engineer and other involved parties. Thus, FMI eases the transition to simulation for non-experts [4].

Simulation is an integral part of a "Digital Twin", and performing it through the use of the FMI not only hides the unnecessary complexity from the user [4], but also results in extra benefits:

- Different types of simulation models, such as Simulink models, the FEA (finite element analysis) simulations, etc., can be integrated and used together [4].
- The user does not need to have expertise in all modelling tools and to meticulously model the functionality and all the interactions among models in order to run the simulation. The simulator (e.g. Multiscale Experiment Creation) performs this job [4].

3.3 Tools and Frameworks

In this section, tools and frameworks that are used and created for the implementation of a "Digital Twin" are studied. The digital twin interface has to reside in an environment that facilitates the integration and utilization of other tools and resources. Therefore, a PLM platform was chosen as the hosting environment where executing simulation and accessing simulation results as well as other resources are handled through web services.

3.3.1 PLM Platform

3DEXPERIENCE is a PLM platform introduced by Dassault Systèmes[14] with the aim of expanding digital and smart product development beyond product lifecycle management. This platform intends to connect the large community of contributors in product development and task management from marketing to sales to engineering. This platform expand the 3D-based digital engineering to other engineering domains. [29]

This platform can be accessed both in the cloud or local installation. Representation of platform capability is specified by a user interface that maintains access to 3D design, analysis, simulation, and intelligence software applications through quadrants of a compass (Figure 16). Furthermore, collaborative and interactive environment of the platform facilitates accelerated report generation and enables distributing and reusing simulation models and results. [29]

Other important features of this platform are extendibility and connectivity. As the platform is in the cloud, third party applications can be readily integrated into the platform and the required data being queried from the cloud. The demo application for the "Digital Twin" is also developed on this platform in order to make use of other applications and platform resources.

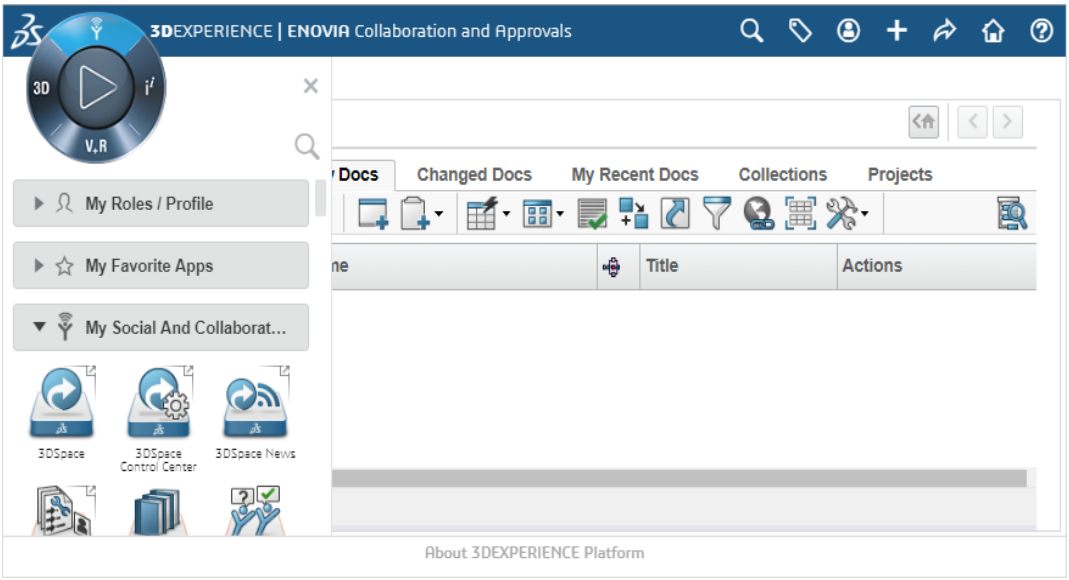

Figure 16. *3DEXPERIENCE platform. Quadrants of the compass on top left, give access to various applications.*

3.3.2 Multiscale Experiment Creation

"Multiscale Experiment Creation" is the name of an environment for creating a process, importing FMUs and running the simulation. Once the FMUs are imported to the process,

[14] https://www.3ds.com

block components of the corresponding simulation model are constructed in the application canvas with the corresponding input and output ports (Figure 15). These ports should be connected correspondingly in order to produce appropriate results upon the execution of simulation model.

3.3.3 Web Services

The web is used to access documents, multimedia and other web resources using unified resource locators (URL) via a web browser. The web can also be used for communication among machines by using web services. There are two general mechanisms for implementing web services: representational state transfer (REST) and simple object access protocol (SOAP). Both mechanisms communicate through the web service description language (WSDL) with some infrastructural differences. [30]

- REST services manipulate web resources over the hypertext transfer protocol (HTTP) through GET, POST, PUT or DELETE methods in order to send a request message from the client to the server. A response message from the server is in the format of XML or JSON, and it is usually requested along with a request message by the client. [30]
- SOAP services use a remote procedure call (RPC) to interact with web resources. In the RPC style, an HTTP URL and a POST method are used only to get the incoming and returning SOAP calls through. The SOAP protocol is included in the XML body and adds a layer of complexity and dependence on XML. [30]

In order to retrieve the resources from the PLM platform, RESTful services are employed. The RESTful services are an API defined by Java for developing web service applications. This choice has been made due to several reasons:

- REST, allowing a variety of data formats such as JSON has some advantages over SOAP where only supports XML. These advantages comprise faster parsing, better support for browser and higher performance.
- The PLM platform comes with Java web service project templates along with libraries. There are also generic methods appended to the project to facilitate the web service creation.

Before creating a web service, a Java object for the resource needs to be generated, next, a method is created to handle the object and last, a Java web service with a method, path and query parameters employs the method to retrieve the requested resource from the PLM platform database. As an example, in order to access simulation data, first the simulation activity should be explored. To find all the simulation processes referenced by a Test Case, first a Test Case object is created, then the method (Program 2 in Appendix A) is generated to extract the object ID of all the simulation processes.

The method shown in Program 2 in Appendix A, finds all the simulation object IDs referenced by a Test Case by using the Test Case object ID. Then the simulation process IDs are appended to a list. The web service that implements this method to handle the request is shown in Program 3 in Appendix A. The GET method is used and a unique path for this purpose. The web service, by means of the method in Program 2 in Appendix A, connects to the data base and handles the request.

Once the web services are created, the project folder is exported to a Java archive (JAR) file and installed on the server. Therefore, requests to the generated paths with appropriate methods and query parameters return the desired response.

3.3.4 Digital Twin Interface

As mentioned in section 3.3.1, one feature of the PLM platform is extendibility. This platform supports development and integration of third party applications. Thus, the application to demonstrate the "Digital Twin" for hybrid installations resides in this platform to leverage the built-in features of the platform.

The digital twin interface for hybrid installations is developed as a web application with a graphical user interface (GUI) to validate the theoretical study of the "Digital Twin" and reveal its benefits and feasibility in practice. The interactions between the digital twin interface and resources in the platform are handled either via web services or by direct interaction, such as drag and drop of documents.

For instance, the integration of the digital twin interface with the "multiscale experiment creation tool". It assists with creation of different configurations of the system by manipulating the variables and abstractions. Therefore, the integration generates a virtual copy of the product and facilitates the instance reasoning and customization of a product.

The digital twin interface aims to elucidate the key features of applying systems engineering and digitalization to a hybrid power module lifecycle while maintaining simplicity and exploiting available resources. The development and deployment of the digital twin interface is more thoroughly explained in chapter 4.

4. IMPLEMENTATION

In this chapter, the implementation of a "Digital Twin" as a demo interface is presented. First, the development and deployment of the application is described more in detail. Afterwards, use cases defined in chapter 3 are validated in order to demonstrate the results. In the end of this chapter, the results of the work and implementation of the digital twin application are presented.

4.1 Development and Deployment

In this section, the development and deployment of the application is delved into. First, software development lifecycles (SDLC) are briefly explained and the adopted approach is identified. Next, the initial design of the web application in the form of sketches and prototypes is presented. Afterwards, the web application development technologies utilized during the development phase are specified. Next, the web service calls and query structure for retrieving the needed data from the platform database and the integration of the application with external sources are described. Last but not least, the deployment of the application on the platform is explained.

4.1.1 Software Development Life Cycle

The SDLC is a framework that aims to structure the development of an application from the early conceptualization phase until its deployment and maintenance. The SDLC approach has numerous similarities with system lifecycle definition since many of the systems engineering principles have been borrowed and adopted from software development. [31]

An SDLC framework is comprised of a set of models that describe the steps followed in application development. These models can be categorized in three major groups: linear, iterative, and a combination of linear and iterative. A linear model is sequential, which means that the next stage will not initiate before the completion of the previous stage. In an iterative model, all the stages are revisited at least once more. A combined model denotes that an iterative model can be halted at a certain stage. [31]

Based on the broad categories above, many models have been recognized. The most popular ones are categorized as the waterfall model, incremental model, V model, spiral model, rapid application development, and agile model. The waterfall model, also known as the cascade model, suggests software to be developed in stages. Royce[15] modified this model and introduced a feedback loop so that each preceding stage could be revisited.

[15] Winston Royce, 1970

The incremental model that is an iterative waterfall starts with a simple implementation of software and evolves by iterations. The V-model, introduced by the INCOSE, is basically similar to that of system lifecycle explained in section 2.1.2. In the spiral model, software development repeatedly passes through iterations. In each spiral a prototype is built, verified against requirements and validated through testing. The rapid application development is a methodology that uses minimal planning for prototyping. In the agile model, application releases are arranged in small time frames, and it ensures that an operating version of the application is always delivered in each iteration. The agile model very well suits small projects and full-stack development. [31]

For development of a digital twin application, an agile model of the SDLC approach is chosen. According to the definition given for the agile model, the timing is arranged so that, in each iteration, a functional application capturing small incremental changes or new features is released and continuously evolved in the following iterations based on feedback and testing.

4.1.2 Design

The conceptualization and initial planning of the project were handled in meetings with a system analysis supervisor[16]. Based on the requirements defined initially and in order to create an insight into the final solution, prototypes of the application were sketched and shared with stakeholders. Figure 17 illustrates a sketch of the simulation engineer's dashboard along with the related information.

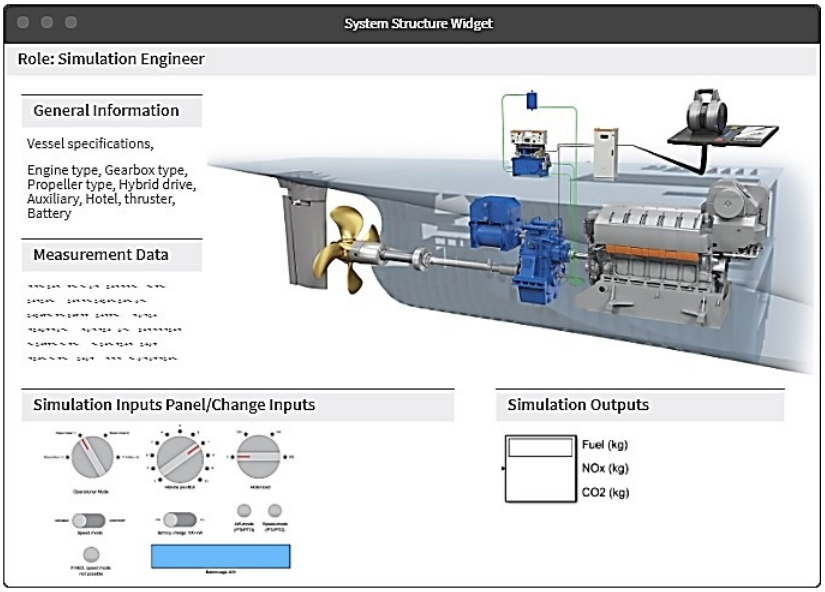

Figure 17. *A sketch of simulation engineer's dashboard.*

[16] Juho Könnö, Wärtsilä Finland Oy

Similarly, sketches for all the use cases, scenarios and actors are created and interactions of each user with the application identified based on the sequence diagrams.

4.1.3 Web Application Development Technologies

Web application development is composed of client side and server side programming. Client side or front-end programming is interpreted by a web browser and interfaces with the user accessing application. Server side programming, on the other hand, is in the back-end and not directly accessible by the user. However, server side programming is used to communicate with the web server and exchange data.

Among numerous technologies for development of a digital twin application, a limited number of web technologies were used for the demo application in this study in order to maintain agile and straightforward development of an application within the scope of this thesis (See Figure 18).

In the client side programming, hypertext markup language (HTML), cascading style sheets (CSS), and JavaScript are the main technologies and backbone of the application interface. Html is used for representing the content of the web application. CSS does the styling, and JavaScript delivers the functionality.

JavaScript libraries are used to ease the application development through providing ready-made modules. jQuery[17] simplifies the client side scripting. React is another library that facilities user interface creation though rendering the application components via client side scripting. Plotly JS[18] is a D3[19]-based (data driven documents) library that assists visualizing the data. Similarly, there are CSS libraries that facilitate styling the application user interface. Bootstrap[20] and Foundation[21] are used for that purpose.

Node JS[22] is a JavaScript framework that supports server side scripting through JavaScript. MongoDB[23] is a free and open-source NoSQL database program. As a part of this project, a Node JS application and MongoDB cloud database are created to store and retrieve some data. Communication between the digital twin application and the Node JS application is maintained by Socket.IO that is a JavaScript library for webSocket.

[17] http://jquery.com
[18] https://plot.ly/javascript
[19] https://d3js.org
[20] https://getbootstrap.com
[21] https://foundation.zurb.com
[22] https://nodejs.org
[23] https://mongodb.com

Figure 18. *Major web development technologies and frameworks used for digital twin application development.*

4.1.4 Coding, Testing, and Integrating

Once the application architecture and use cases are defined, the process of application creation begins. HTML, JavaScript, and CSS files form the basis of the web application. In the HTML file, the application metadata, the CSS style sheets, and scripts are declared. The script tag in HTML references Require JS[24] that is a JavaScript module loader. Therefore, one Script tag handles the loading of all modules and improves the speed and quality of code.

On the client side, object oriented JavaScript is used and use cases are implemented in a modular manner to support reusability and ease of maintenance. The JavaScript code forms the major portion of coding. Thanks to React JS, most of the HTML code is also implemented in the JavaScript code and components are initiated using a Foundation stylesheet and rendered to the corresponding HTML placeholders.

Node JS application is a minimal solution that is merely used as a bridge between the digital twin application and MongoDB cloud database. Implementation of Node JS application is also handled through JavaScript coding.

Furthermore, in order to transfer data between a web server and web browser, XML Http Request (XHR) is used. XHR API is formed based on the web service previously created (section 3.3.3). The PLM platform comes with a library that assists with web service calls. Program 4 in Appendix A shows how a web service call is structured. Similar to XHR API, it is possible to specify the request method, query parameters, and data type.

[24] http://requirejs.org

Code testing and debugging is a crucial part of application development. Errors in the syntax and logics of the program are inevitable. Searching for the errors, modifying them and testing the application requires significant time and effort. This process is handled in the browser console.

4.1.5 Deployment on Platform

Once the application is deployed on the web server through a built-in mechanism in the PLM platform, an icon for the application shows up in the platform application quadrant pointing to the URI of the application on the web server. An embedded application inside the platform is called a “Widget”. The platform supports execution and communication among multiple widgets on the dashboard.

The deployed application is further tested to assure that it works flawlessly inside the platform in terms of accessing the PLM platform inherit libraries. Moreover, although the communication with external application, e.g. the Node JS application, is maintained through the web socket, the access could still be hindered due to security issues. Such issues are handled via incorporating a self-signed certificates for external applications.

4.2 Results

The main goal of a “Digital Twin” is to increase an insight into the target system. Interactions of a human with the application can be modeled as human-in-the-loop where the outcome and results of the process are affected by human choices. Simulation results and data out of isolation can be used for various applications and simulation-based approaches. Simulation-based validation and optimization are some examples [32].

4.2.1 Task-centered Product Development

The term “task-centered” responses to the question of who is going to access the system to do what. The industry terminology for this term is “task and user analysis”. Task analysis refers to the fact that the systems must be able to handle what it is intended for. User analysis, on the other hand, refers to the appropriate distribution of tasks to the system users so that the relevant task and information is correctly routed to the intended system users.

One of the main objectives of digital twin application is communication among the involved players and value co-creation. This justifies the task and user analysis in the design and implementation phase of the application. Users of the application must be carefully identified based on the tasks that have to be fulfilled by adhering to the systems engineering lifecycle approach explained in section 2.1.3.

In order to deliver tasks and information appropriately to users, each user must be first clearly characterized and identified. Defining the user in the design phase is thus of a great importance. The users of the digital twin application were characterized in section 3.1.3. The users are narrowed down to a simulation engineer, customer, and product manager. First, on logging in to the application, the user is selected. Figure 19 shows the login page of the digital twin application. The user, logs into the digital twin application with the user credentials and selection of module and vessel types.

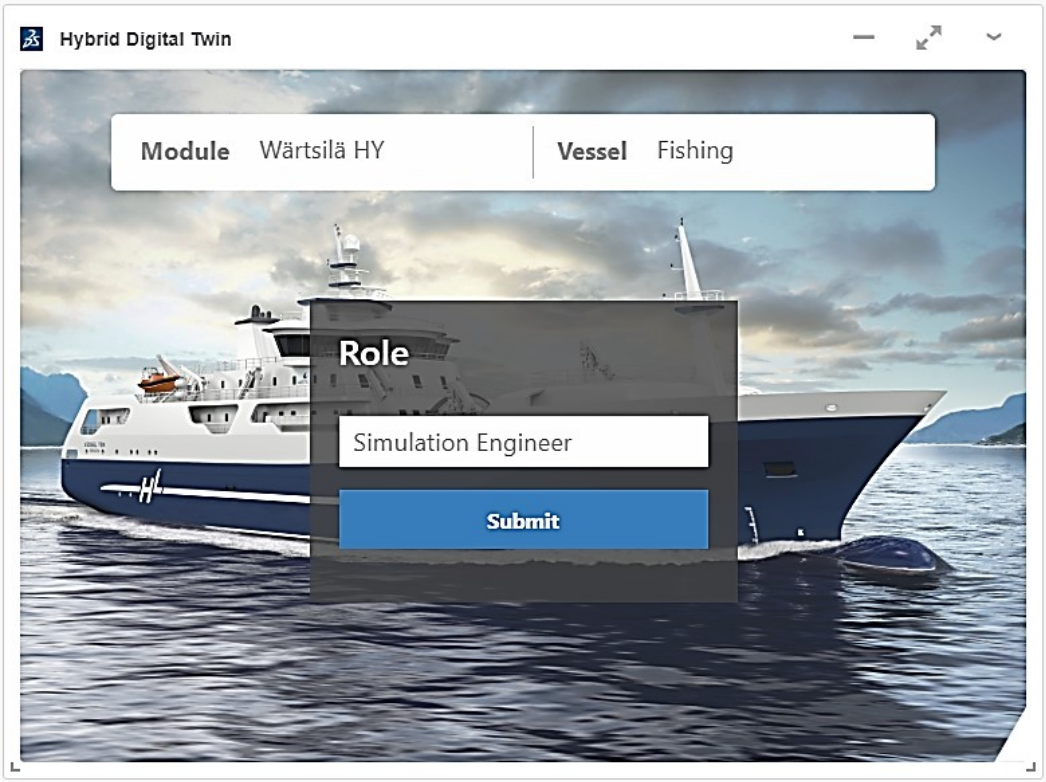

Figure 19. *User, module and vessel type are identified when logging in to the application.*

In the next step, after logging in to the application, corresponding tasks and information interface with the user. This is handled by implementing the application architecture defined in section 3.1.3. Based on the definition given for task-centered product development, tasks and information rendered for each user have to amount to a concrete pattern for product lifecycle management. This pattern, in the course of this study, is the “V” lifecycle model of systems engineering (see Figure 20).

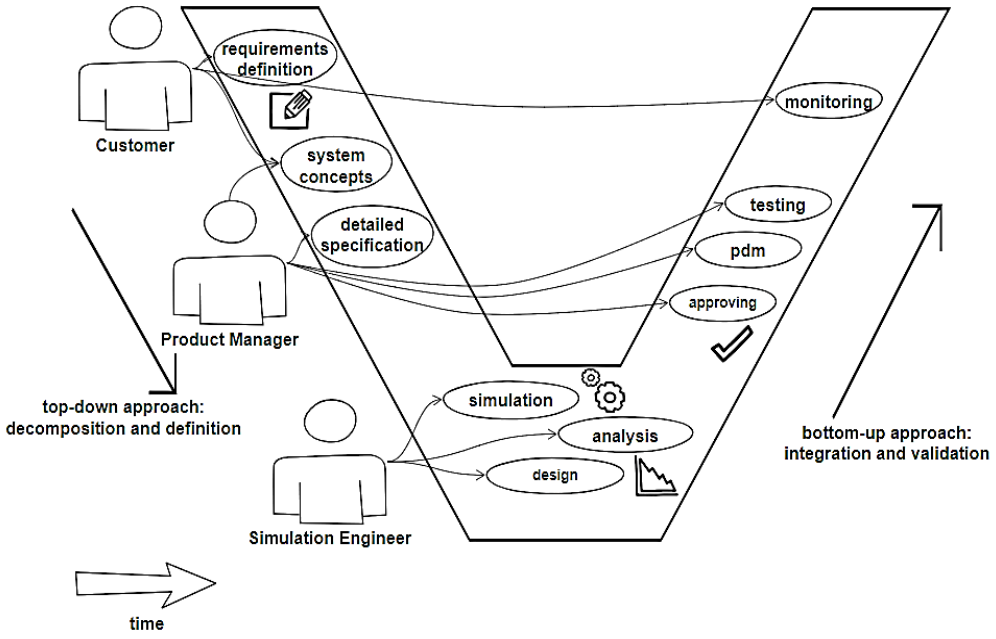

Figure 20. *Simplified "V" lifecycle model of systems engineering in task-centered product development.*

Based on the definition given above and initial design considerations, the results of implementing task-centered product development can be outlined. Customer, as the starting point in the top-down approach of the simplified "V" lifecycle model of systems engineering (Figure 20), defines the product and requirements from the customer's point of view while dismissing the technical design matters. Figure 21 shows the system requirements panel in the digital twin application that is merely accessible by customer.

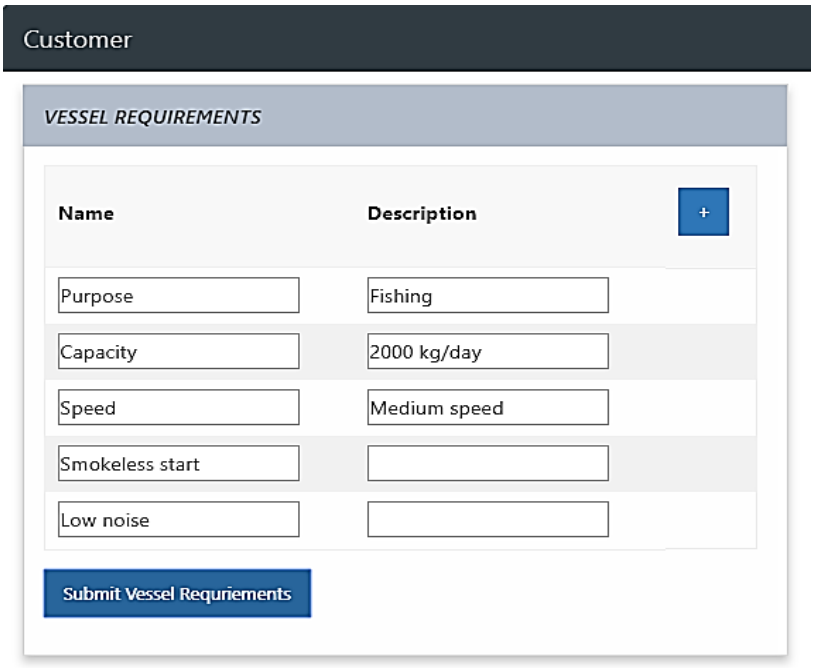

Figure 21. *System requirements panel personalized for the customer.*

The product manager, as another application user, acts as a connection between the customer and the simulation engineer. This is also depicted in Figure 20. For instance, requirements previously created by the customer should then be translated to the technical system requirements. The product manager takes care of integrations as such. The translation of the customer's needs to requirement specifications is handled though a built-in application in the PLM platform. Afterwards, the requirement specification is imported to the digital twin application by dragging it from the platform and dropping it into the specified area in the product manager's dashboard (see Figure 22).

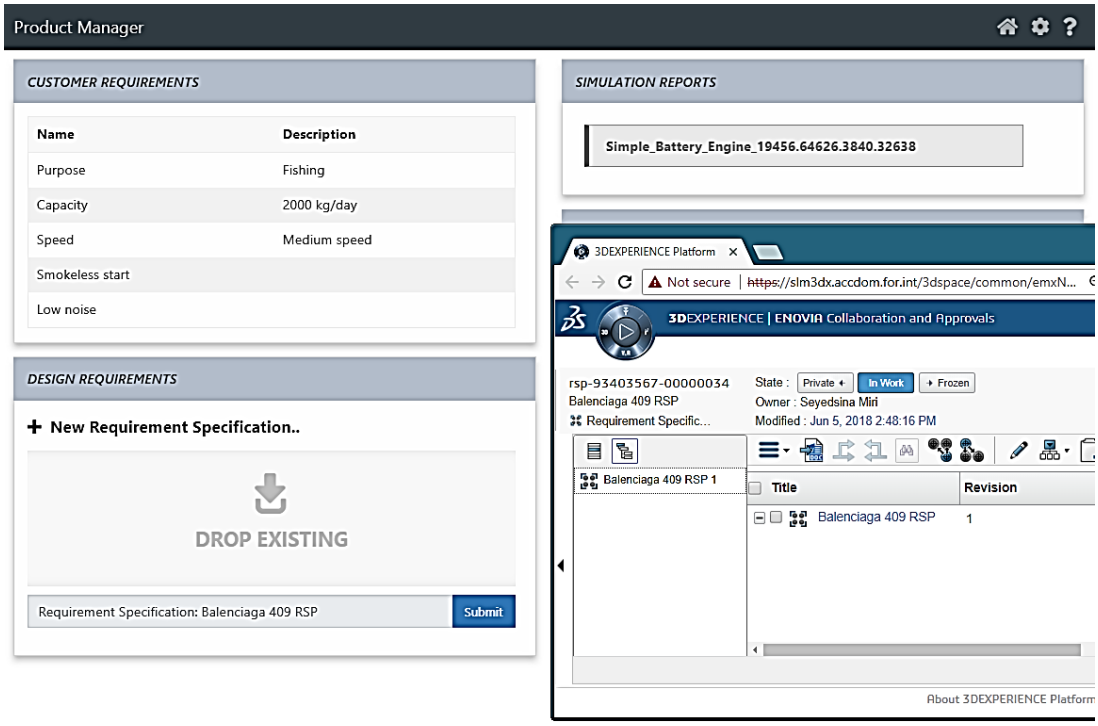

Figure 22. *Requirement specification creation in platform and its integration with application.*

In the lower layers of the simplified "V" lifecycle model of systems engineering (Figure 20), system design and simulation are dominant chores that have to be dealt with according to the requirement specification generated by the product manager. In the digital twin application, this layer is simplified and converged to the simulation engineer actor. Task-centered product development necessitates a set of tasks to be fulfilled by the simulation engineer.

The simulation engineer can browse the requirements and, based on the requirement specification generated by the product manager, creates a simulation or uses an existing simulation in the platform to attach to the requirement specification. Simulation-based validation and product requirements analysis are among other major tasks designated to the simulation engineer. They will be thoroughly discussed in section 4.2.2 and 4.2.3 respectively.

At this point, all the stages of the simplified “V” lifecycle model of systems engineering in a top-down approach are paved. Then, the tasks have to be appropriately distributed to the users in the bottom-up approach (Figure 20). The simulation engineer generates a report based on simulation and requirement validation along with the simulation configuration chosen. The product manager approves one of the simulation configurations based on the report from the simulation engineer. Customer in the meanwhile, monitors the status of the product.

Besides the core tasks mentioned so far, there are other tasks and information interfacing with the application users that further assist them with product development. These tasks are further elaborated in the following sections.

4.2.2 Simulation-based Validation

Once the simulation is successfully executed on the platform, the simulation results in the form of the csv are recorded into the platform file system. By querying this data via the REST API, a link to the simulation results is generated. By accessing the data and parameterizing them, arrays of data can be utilized for different analyses.

On selecting of a simulation configuration by simulation engineer, generic plots of all the simulation data (mostly consist of inputs and outputs of the blocks) are created on one panel. Figure 23 illustrates some graphs of engine parameters with respect to time.

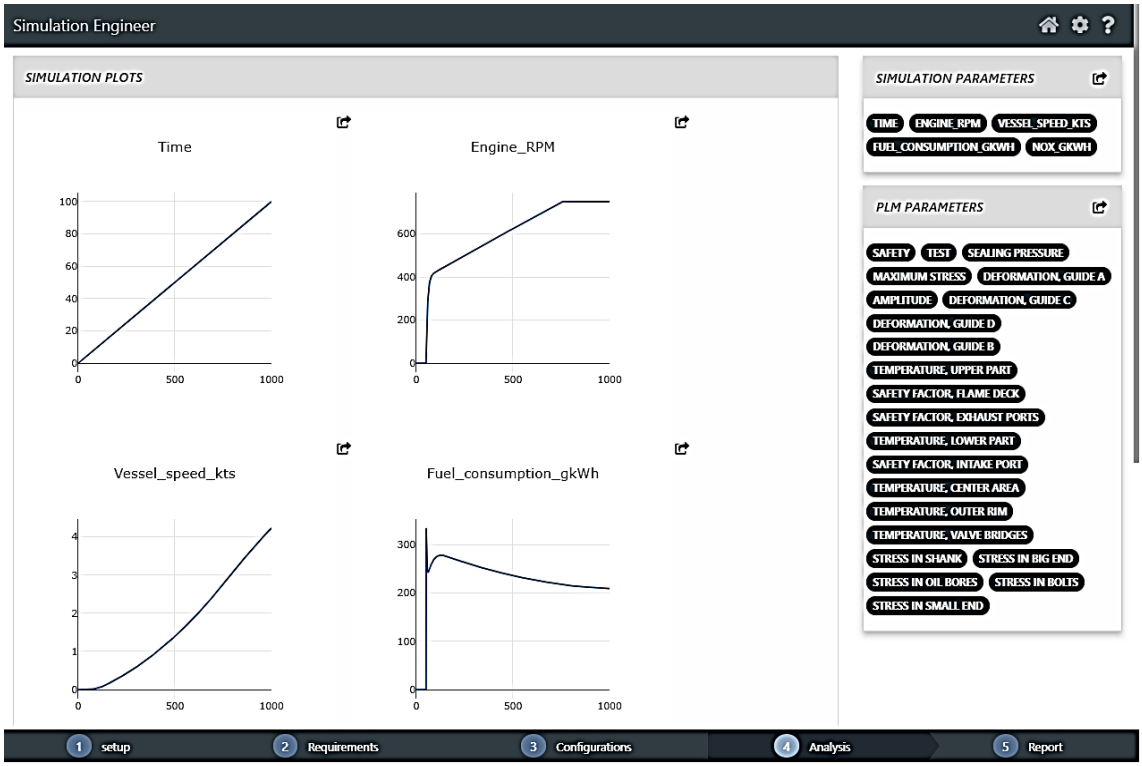

Figure 23. *Generic plots of simulation results of a selected configuration.*

Moreover, the test data from product components, e.g. engine, are hardwired to the application. This is of significant importance in reasoning of the simulation results. Therefore, a custom and an interactive plotting tool are also developed in a separate panel where the simulation data and test results can be visualized, compared, and validated. Figure 24 shows an example where the simulation data is validated with its corresponding test data by using the interactive plotting tool developed in the digital twin application. In this plot, the result of the rotational speed of an engine from a simulation is compared with that of the test results.

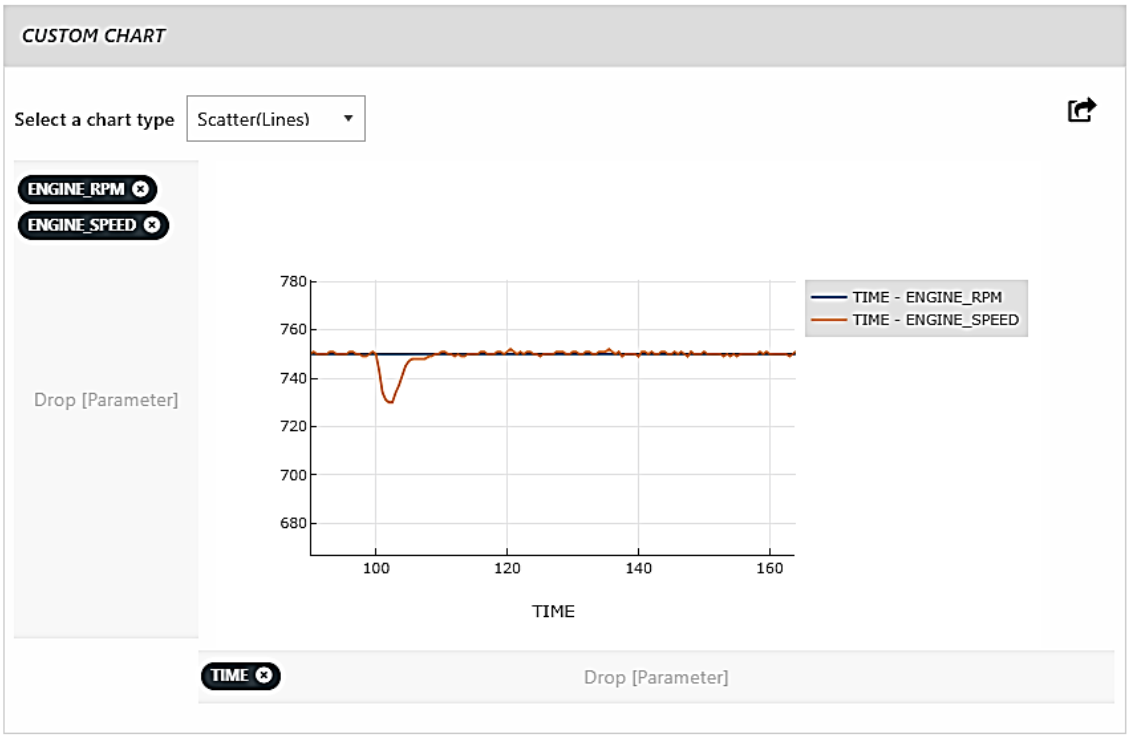

Figure 24. *Validation of simulation data with corresponding test results.*

4.2.3 Product Requirements Analysis

System requirements are generated as PLM parameters during the systems requirements definition phase. PLM parameters are global product parameters. These requirements are structured in the platform so that the requirement specification forms a tree structure with chapters and requirements under them that further expand to test cases. Each test case may reference to a simulation template and a number of PLM parameters. Each test case is further expanded to a number of test executions that each represent an individual run or execution per design iteration. Thus, each test execution may reference to a simulation process indicating different instances of the simulation template referenced by the parent test case.

In order to visualize the abovementioned functionality in the application, first the REST API (explained in section 3.3.3) for the given purpose is created and deployed on the

server. Afterwards, thorough XHR API explained in section 4.1.3, the needed resources and objects are queried and visualized in the application.

Figure 25 illustrates how a requirement specification is structured. In this specific example, engine concept validation as the requirement specification has multiple chapters underneath. The cylinder head chapter is expanded to a number of requirements namely safety factor, valve guide deformation, flame deck temperatures, and Gasket sealing pressure. Further expanding of one of the requirements leads to test cases, each with corresponding test execution, PLM parameters and possible referenced simulations.

On the right side, there is a panel that represents more detailed information about each object underneath the requirement specification and is represented upon clicking on each object. The information on this panel is based on the metadata that is automatically generated while creating or maintaining objects on the platform.

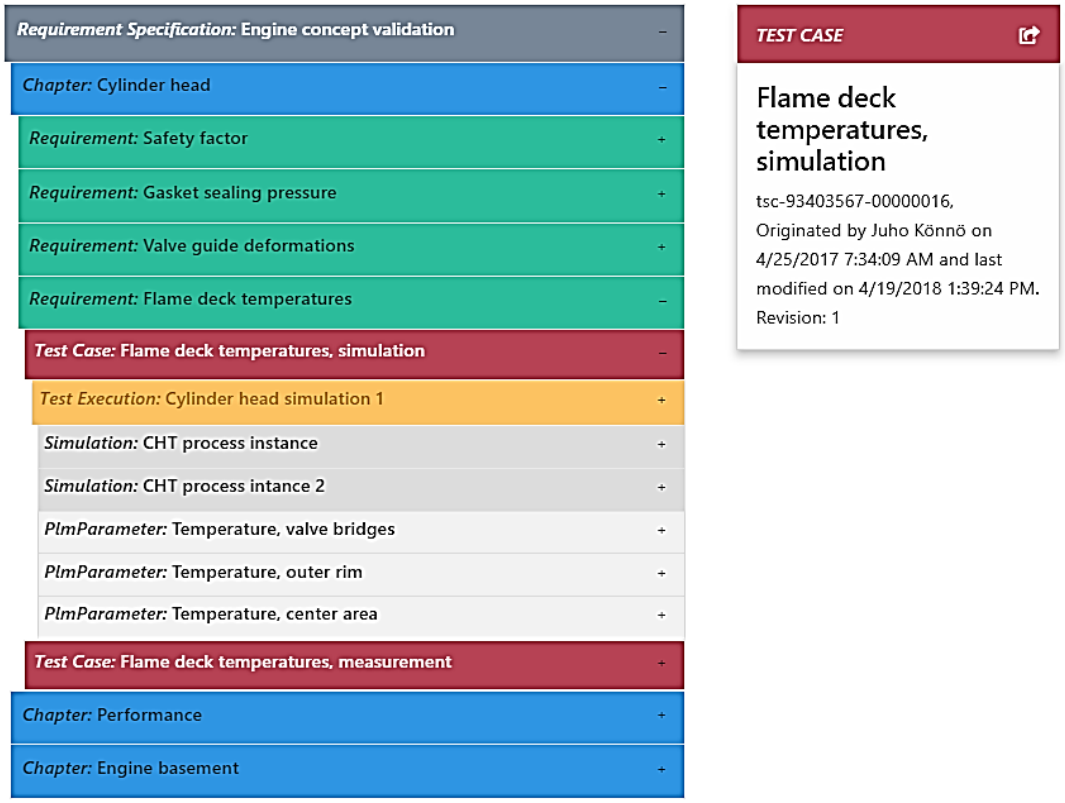

Figure 25. *Tree structure for a requirement specification on the left. Object metadata panel on the right.*

In the digital twin application, it is assumed that the requirement specifications are created by the product manager in the platform based on the customer's needs. Similarly to simulation-based validation, requirements analysis is one of the tasks of the simulation engineer.

Product requirements data stored as PLM parameters are then queried using the REST API, and their values (usually specified as minimum and maximum allowed values) are

parameterized. Thus, similarly to simulation data validation, the product requirements can be compared with the simulation results. Figure 26 show the visualization of one of the requirements that can be used for validation of simulation results. Requirements validation for the simulation results is also achieved in a similar manner as validation using test data depicted in Figure 24.

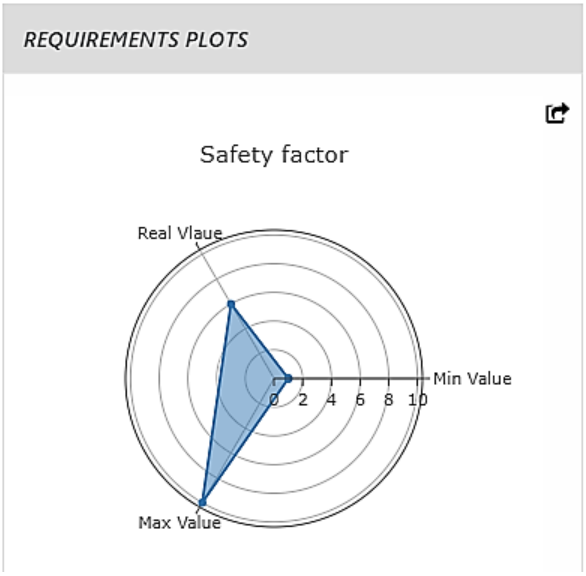

Figure 26. *Product requirements visualization.*

4.2.4 Business Value Prospects

One of the main objectives of implementing the digital twin application is the convergence of information that expedites decision making in business functions and sales. Among the wide range of business opportunities, product data management and resource planning are implemented in digital twin application. Following, some of the business value prospects are outlined:

- PDM is an existing software and database that contains comprehensive information about different products, their detailed bills of materials, and their real-time status. One of the crucial tasks of the product manager is product data management and resource planning. The digital twin application, by maintaining communication with PDM software, allows the product manager to query the product data and make the selection for the bill of materials for the product based on the reports from the simulation engineer. Figure 27 illustrates a panel where the product manager queries the data regarding different engines.

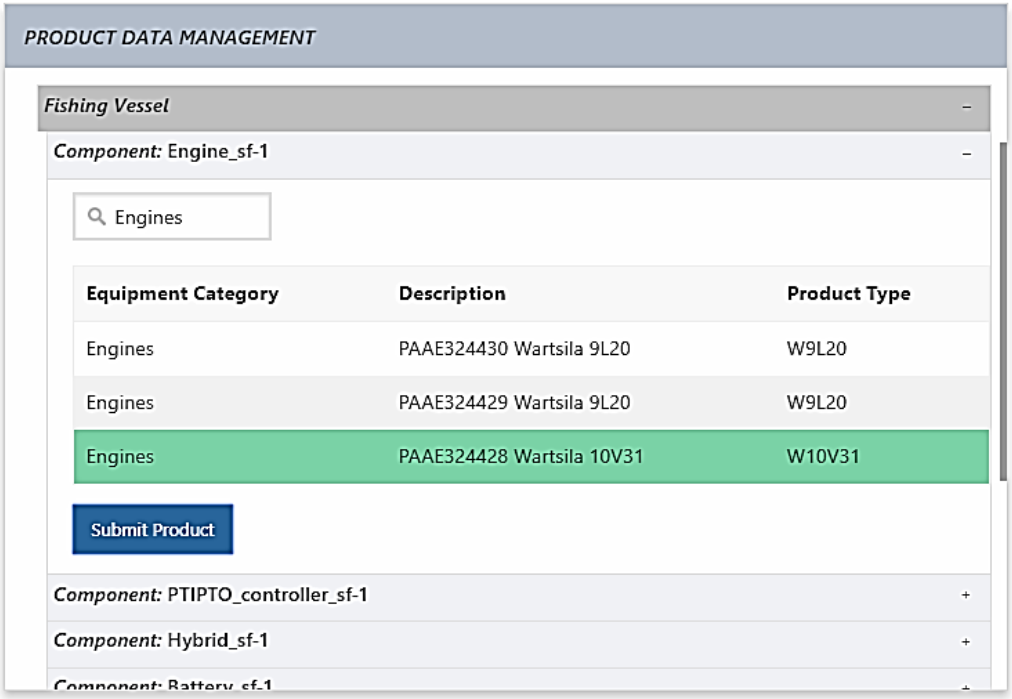

Figure 27. *Integrating PDM with the digital twin application.*

- Service scheduling, condition based monitoring and predictive maintenance are among other business value prospects that can be exploited from the digital twin implementation. Since the focus of this study is solely on the early stages of product development and no operating data is yet available, its implementation in the digital twin application is restricted to a panel where the product manager can manually add maintenance records and service schedules.

5. CONCLUSION

This chapter concludes the theoretical study and implementation of the digital twin application for hybrid installations. First, accomplishments based on the goals set initially are reviewed. Afterwards, challenges and limitations in implementation of the application are narrowed down. Finally, the possible future work and modifications are outlined.

5.1 Accomplishments

Based on the results of theoretical research, the "Digital Twin" can be inferred as a general system model that is initiated with the products conceptualization and constantly evolves with the product lifecycle.

Throughout the theoretical study of the "Digital Twin" concept and implementing it practically, the goals initially set have been pursued. A demo application for digital twin embedded in the PLM platform leverages built-in platform features alongside integration with other services and introduction of new features. The following main achievements are noted:

- Flawless integration and communication of the digital twin application with the existing PLM platform data through web services.
- Applying systems engineering principles and model-based systems engineering to the product development and implementing it in a simplified manner in the digital twin application.
- Assessing and analyzing simulation results and product requirements. Reasoning and validation of these data with respect to acquired test results.
- Supporting the business and sales functions by facilitating the bill of materials selection through connection of the digital twin application with the PDM.

5.2 Challenges and Limitations

Undoubtedly, development of ideas, concepts and methods cannot be handled without coping with any challenges and limitations. The majority of challenges in implementing the digital twin application was faced in integration with the platform and communication with its resources. Moreover, the execution of simulation inside the platform was not carried out without challenges, and many sessions were dedicated to fixing errors together with the simulation engineer and platform support.

Apart from limitations set in the beginning of the study, another limitation arose on the way of implementing the application. In the initial plan, the FMU simulator was supposed

to be embedded inside the digital twin interface in order to facilitate dynamically simulating the product configurations based on the product requirements specifications. However, lack of an open API for the simulator hindered the plan and instead the "multiscale experiment creation tool" was decided to be used in companion with the digital twin interface.

5.3 Future Work

The concept and implementation of the digital twin application could be extended to take account of a wider scope. A number of future work possibilities are outlined here:

- A future work could focus on other stages of product lifecycle, such as manufacturing and servicing. The current solution merely covers the system design phase.
- The implementation could extend to other types of modules and productions than the hybrid power modules.
- Another future work could look for methods that automate some or all use cases within the product lifecycle stages. For instance, instead of manually creating product configuration of product, a method could dynamically generate product configurations based on standardized product requirement specifications. Consequently, simulating the created configurations and validating them.
- An important aspect of the "Digital Twin" concept is the real-time monitoring of the system and retrieving the sensory data. This aspect is ignored in the course of this study as the hybrid power module production has just been initiated. Thus, a future work could take such use cases into account.

REFERENCES

[1] M. E. Porter and J. E. Heppelmann, “How smart, connected products are transforming competition,” *Harv. Bus. Rev.*, vol. 92, no. 11, pp. 64–88, 2014.

[2] E. Glaessgen and D. Stargel, “The digital twin paradigm for future NASA and US Air Force vehicles,” in *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference 20th AIAA/ASME/AHS Adaptive Structures Conference 14th AIAA*, 2012, p. 1818.

[3] R. Rosen, G. Von Wichert, G. Lo, and K. D. Bettenhausen, “About the importance of autonomy and digital twins for the future of manufacturing,” *IFAC-PapersOnLine*, vol. 48, no. 3, pp. 567–572, 2015.

[4] M. Schluse, L. Atorf, and J. Rossmann, “Experimentable digital twins for model-based systems engineering and simulation-based development,” in *Systems Conference (SysCon), 2017 Annual IEEE International*, 2017, pp. 1–8.

[5] M. Peruzzini, M. Germani, and C. Favi, “Shift from PLM to SLM: a method to support business requirements elicitation for service innovation,” in *IFIP International Conference on Product Lifecycle Management*, 2012, pp. 111–123.

[6] B. Schleich, N. Anwer, L. Mathieu, and S. Wartzack, “Shaping the digital twin for design and production engineering,” *CIRP Ann.*, vol. 66, no. 1, pp. 141–144, 2017.

[7] W. R. Adrion, “Research methodology in software engineering, Summary of the Dagstuhl Workshop on Future Directions in Software Engineering, W,” *ACM SIGSOFT Softw. Eng. Notes*, vol. 18, no. 1, 1993.

[8] D. D. Walden, G. J. Roedler, K. Forsberg, R. D. Hamelin, and T. M. Shortell, *Systems engineering handbook: A guide for system life cycle processes and activities*. John Wiley & Sons, 2015.

[9] R.I. Faulconbridge, M.J. Ryan, "Systems Engineering Practice”, *Argos Press, Canberra,* 2014.

[10] J. A. Estefan, “Survey of model-based systems engineering (MBSE) methodologies,” *Incose MBSE Focus Gr.*, vol. 25, no. 8, pp. 1–12, 2007.

[11] C. Bock, “SysML and UML 2 support for activity modeling,” *Syst. Eng.*, vol. 9, no. 2, pp. 160–186, 2006.

[12] R. Silhavy, P. Silhavy, and Z. Prokopova, “Behavioral modeling in system engineering,” in *Proceedings of the 13th WSEAS international conference on Automatic control, modelling & simulation*, 2011, pp. 100–105.

[13] C. Zhuang, J. Liu, and H. Xiong, “Digital twin-based smart production management and control framework for the complex product assembly shop-floor,” *Int. J. Adv. Manuf. Technol.*, vol. 96, no. 1–4, pp. 1149–1163, 2018.

[14] S. Boschert and R. Rosen, “Digital twin—the simulation aspect,” in *Mechatronic Futures*, Springer, 2016, pp. 59–74.

[15] M. Abramovici, J. C. Göbel, and H. B. Dang, “Semantic data management for the development and continuous reconfiguration of smart products and systems,” *CIRP Ann. Technol.*, vol. 65, no. 1, pp. 185–188, 2016.

[16] M. Abramovici and A. Lindner, “Providing product use knowledge for the design of improved product generations,” *CIRP Ann. Technol.*, vol. 60, no. 1, pp. 211–214, 2011.

[17] F. Tao, J. Cheng, Q. Qi, M. Zhang, H. Zhang, and F. Sui, “Digital twin-driven product design, manufacturing and service with big data,” *Int. J. Adv. Manuf. Technol.*, vol. 94, no. 9–12, pp. 3563–3576, 2018.

[18] T. D. West and M. Blackburn, “Is Digital Thread/Digital Twin Affordable? A Systemic Assessment of the Cost of DoD’s Latest Manhattan Project,” *Procedia Comput. Sci.*, vol. 114, pp. 47–56, 2017.

[19] M. Schluse and J. Rossmann, “From simulation to experimentable digital twins: simulation-based development and operation of complex technical systems,” in *Systems Engineering (ISSE), 2016 IEEE International Symposium on*, 2016, pp. 1–6.

[20] “Industry 4.0 and the digital twin” [Online]. Available: https://www2.deloitte.com/content/dam/Deloitte/cn/Documents/cip/deloitte-cn-cip-industry-4-0-digital-twin-technology-en-171215.pdf. [Accessed: May-2018].

[21] Wärtsilä Marine Solutions, “HYBRID activities Engine tech. - Info/steering meeting” [Internal], 2018.

[22] Wärtsilä Marine Solutions, “Wärtsilä Solutions for Marine and Oil & Gas Markets”, [Online brochure]. Available: https://cdn.wartsila.com/docs/default-source/marine-documents/segment/brochure-marine-solutions-2017.pdf?sfvrsn=658adc45_34, 2017. [Accessed: May-2018].

[23] S. M. Ross, *A course in simulation*. Prentice Hall PTR, 1990.

[24] MATLAB/Simulink, [Online]. Available: www.mathworks.de/products/simulink/. [Accessed: May-2018].

[25] F. Cremona, M. Lohstroh, D. Broman, E. A. Lee, M. Masin, and S. Tripakis, "Hybrid co-simulation: it's about time," *Softw. Syst. Model.*, pp. 1–25, 2017.

[26] FMI History, [Online]. Available: http://fmi-standard.org/history/. [Accessed: May-2018].

[27] M. Mitterhofer, G. F. Schneider, S. Stratbücker, and K. Sedlbauer, "An FMI-enabled methodology for modular building performance simulation based on Semantic Web Technologies," *Build. Environ.*, vol. 125, pp. 49–59, 2017.

[28] N. Amálio, R. Payne, A. Cavalcanti, and J. Woodcock, "Checking SysML models for co-simulation," in *International Conference on Formal Engineering Methods*, 2016, pp. 450–465.

[29] A. Barth, "3D Experiences–Dassault Systèmes 3DS Strategy to Support New Processes in Product Development and Early Customer Involvement," in *Digital Product and Process Development Systems*, Springer, 2013, pp. 24–30.

[30] J. L. Jensen, A. J. Bohonak and S. T. Kelley, "Isolation by distance, web service," BMC Genetics, vol. 6, (1), pp. 13-13, 2005.

[31] N. B. Ruparelia, "Software development lifecycle models," *ACM SIGSOFT Softw. Eng. Notes*, vol. 35, no. 3, pp. 8–13, 2010.

[32] S. Kadry and E. H. Abdelkhalak, "*E-Systems for the 21st Century: Concept, Developments, and Applications-Two Volume Set"*. Apple Academic Press, Inc., 2016.

APPENDIX A: PROGRAMS

```
<?xml version="1.0" encoding="UTF-8"?>
<fmiModelDescription
  fmiVersion="2.0"
  modelName="Engine_sf"
  guid="{38eb6135-5153-4198-a1a7-8fc7d52e5191}"
  description="S-function with FMI generated from Simulink model
Engine"
  author="Unknown"
  version="1.5"
  generationTool="Dassault Systemes FMI Kit for Simulink, ver.
2.4.0 (MATLAB 8.11 (R2016b) 25-Aug-2016)"
  generationDateAndTime="2017-12-20T06:18:48Z"
  variableNamingConvention="structured"
  numberOfEventIndicators="0">
  <CoSimulation
    modelIdentifier="Engine_sf"
    canHandleVariableCommunicationStepSize="true"
    canInterpolateInputs="true"/>
  <DefaultExperiment startTime="0.0"
    stepSize="0.1"/>
  <ModelVariables>
  <ModelStructures>
</fmiModelDescription>
```

Program 1. *Model description XML defines the model variables and structure.*

```
public ArrayList<SimulationProcessObject> getSimulationPro-
cessesReferencedByTestCase(Context context, TestCaseObject
testCaseObject) throws Exception {
      // Container
      ArrayList<SimulationProcessObject> simulationProcessOb-
jectList = new ArrayList<SimulationProcessObject>();

      // Create Handle to object id
      DomainObject dom = new DomainObject(testCaseObject.get-
TestCaseBusinessObject().getObjectId());
      String relType = Common.RELATIONSHIP_TYPE_REFER-
ENCED_SIMULATIONS;
      String objType = Common.OBJECT_TYPE_SIMULATION;
      StringList objSelectList = new StringList(DomainCon-
stants.SELECT_ID);
      short recurseLevel = 1;

      MapList relBusObjPageList = dom.getRelatedObjects(con-
text, relType, objType, objSelectList, null, true, true, re-
curseLevel, "", "");

      // Loop Map List and create array of simulation process
Object
      for (int ii = 0; ii < relBusObjPageList.size(); ii++) {
      // Get the simulation process object id
      String simulaitonProcessId = ((Map<String, String>)
relBusObjPageList.get(ii)).get("id");

      // Create a simulation process object
      SimulationProcessObject simulationProcessObject = new
SimulationProcessObject(new BusinessObject(simulaitonProces-
sId));

      // Append this test case to the list
      simulationProcessObjectList.add(simulationProcessOb-
ject);
      }
      return simulationProcessObjectList;
}
```

Program 2. *A method to extract all the simulation processes referenced by a Test Case*

```
@GET
@Path("/getSimulationProcessesReferencedByTC")
public Response getSimulationProcessesReferencedByTC(@javax.ws.rs.core.Context HttpServletRequest request, @QueryParam("id") String id) {

  JSONObject output = new JSONObject();
  matrix.db.Context context = null;
  try {
    output.put("msg", "KO");
    JSONArray mySimulationProcessList = new JSONArray();
    try {
      boolean isSCMandatory = false;
      context = getAuthenticatedContext(request, isSCMandatory);
      ENOCSWebServServices.setRoleonContext(context);

      WartsilaMethods wartsilaMethods = new WartsilaMethods();
      ArrayList<SimulationProcessObject> simulationProcessList = wartsilaMethods.getSimulationProcessesReferencedByTestCase(context, new TestCaseObject(new BusinessObject(id)));

      for (SimulationProcessObject simulationProcessObject : simulationProcessList) {
        mySimulationExperienceList.put(simulationExperienceObject.getSimulationExperienceBusinessObject().getObjectId());
      }
      output.put("Simulation Processes", mySimulationProcessList);
        output.put("msg", "OK");

    } catch (Exception e) {
      output.put("msg", e.getMessage());
      e.printStackTrace();
    }
  } catch (Exception e1) {
    e1.printStackTrace();
  }
  return Response.status(HttpServletResponse.SC_OK).entity(output.toString()).build();
}
```

Program 3. *Web service to GET all the simulation process referenced by a Test Case.*

```
WAFData.authenticatedRequest(the3DSpaceUrlService+'/Simulia-
Tools/ExpandObject', {

  'method': 'GET',
  'data': {
    'objectId': oid
  },
  'type': 'json',
  'onComplete': someFunction(data)
})
```

Program 4. *Web service call used for communicating with 3DEXPERIENCE web server.*