



Forschungsbereich
Maschinenbau Informatik
und Virtuelle
Produktentwicklung

Asset Integration and Processing in a Holistic Large-scale Railway Infrastructure Simulation System

Dissertation

Ozan Kugu M.Sc.



TECHNISCHE
UNIVERSITÄT
WIEN
Vienna | Austria



Dissertation

Asset Integration and Processing in a Holistic Large-scale Railway Infrastructure Simulation System

carried out for the purpose of obtaining the degree of
Doctor of Technical Sciences (Dr. techn.)

submitted at TU Wien, Faculty of Mechanical and Industrial Engineering, by

Ozan Kugu

Mat.Nr.: 12131142

under the supervision of

Ao. Univ.Prof. Dipl.-Ing. Dr.techn. Manfred Grafinger

E307 - Institute of Engineering Design and Product Development

Research Unit of Mechanical Engineering Informatics and Virtual

Product Development

Vienna, December 2025

Reviewed by

.....
Univ.-Prof. Dipl.-Ing. Dr.-Ing. Detlef Gerhard

Ruhr-Universität Bochum
Institute for Product and Service Engineering
Digital Engineering Chair

.....
Prof. Dr. Doris Aschenbrenner

Hochschule Aalen
Digital Methods in Production
Carl-Zeiss-Stiftung

This work was supported by Austrian Research
Promotion Agency (FFG) within the framework of the
Rail4Future project (Nr. 35841855).

I confirm, that going to press of this thesis needs the confirmation of the examination
committee.

Affidavit

I declare in lieu of oath, that I wrote this thesis and performed the associated research myself, using only literature cited in this volume. If text passages from sources are used literally, they are marked as such.

I confirm that this work is original and has not been submitted elsewhere for any examination, nor is it currently under consideration for a thesis elsewhere.

I acknowledge that the submitted work will be checked electronically-technically using suitable and state-of-the-art means (plagiarism detection software). On the one hand, this ensures that the submitted work was prepared according to the high-quality standards within the applicable rules to ensure good scientific practice "Code of Conduct" at the TU Wien. On the other hand, a comparison with other student theses avoids violations of my personal copyright.

Vienna, 9. January 2026

City and Date



Signature

Vorwort

Die Dissertation wurde zur Erlangung des Titels Doktor der Technischen Wissenschaften (Dr.techn.) erstellt. Die Arbeit zeigt und beschreibt wie verschiedene Simulationsmodelle und -daten, die zu den Subsystemen eines Eisenbahninfrastruktursystems (Bsp. Zug, Schiene) gehören, in einer digitalen Zwillingsplattform, heißt Rail for Future (R4F), im Rahmen des Rail4Future Projekts (2021-2024) integriert und bearbeitet werden. Dafür wurden Kenntnisse über bestimmte Schlüsseltechnologien, die spezifisch für die Integrations- und Bearbeitungsaufgabe ausgewählt wurden, benötigt. Außerdem mussten Methoden entwickelt werden um die angemessene Anwendung der Schlüsseltechnologien in der R4F Plattform zu schaffen.

Die Dissertation entstand während meines Doktoratsstudiums der technischen Wissenschaften an der TU Wien in der Zeit von März 2022 bis Dezember 2025. Im Zuge des Studiums und Rail4Future Projekts war ich immer in Informationsaustausch mit meinen internen Kollegen und externen Projektpartnern, was ich wichtig für meine Motivation, individuelle Personalentwicklung und anschließend Fertigstellung der Dissertation finde.

Erstens möchte ich mich bei Ao. Univ.Prof. Dipl.-Ing. Dr.techn. Manfred Grafinger für seine Hauptbetreuung, Kompetenz und besonderes Interesse an meine Arbeit bedanken. Außerdem bedanke ich mich sehr bei meinen Assistentenkollegen und Studenten, die mir erheblich geholfen haben. Ich leite meine Danksagungen auch zu Univ.-Prof. Dipl.-Ing. Dr.-Ing. Detlef Gerhard und Prof. Dr. Doris Aschenbrenner für ihre hervorragende Begutachtung meiner Arbeit weiter. Zu guter Letzt gebe ich meine Danksagungen zu meiner Familie und besten Freunden weiter, die mit mir gerne gesunde, lustige und nützliche Zeiten verbringen. Daher darf ich sie nie in meinem Leben ignorieren.

Ich wünsche euch viel Spaß beim Lesen der Dissertation!

Wien, 27. December 2025

Ozan Kugu M.Sc.

Kurzreferat

Eisenbahnen haben einen relativ hohen Anteil am öffentlichen Verkehr und am Güterverkehr, denn sie sorgen für klimaneutrale, geldsparende und erlebnisreiche Mobilität im Alltag. Neben ihrem Marktanteil spielen der Betrieb und die Instandhaltung der Eisenbahninfrastruktur eine wichtige Rolle bei der Aufrechterhaltung der Nachhaltigkeit der hochkomplexen und aufwendigen Eisenbahnsysteme. Zu diesem Zweck wird die Eisenbahninfrastruktur tendenziell digitalisiert, was die vorausschauende Überwachung und Wartung verschiedener Teilsysteme des Systems wie Schienenfahrzeuge, Brücken, Tunnel, Gleise und Weichen in einer virtuellen Umgebung ermöglicht. Seit einigen Jahren wird die Technologie des digitalen Zwillings (DT) als ein potenzieller Kandidat zur Verbesserung der Digitalisierungsaufgabe im Eisenbahnsektor angesehen. In diesem Fall interagiert das reale System mit seiner virtuellen Variante, wobei die beiden Systeme Modelle und Daten sowie Simulationen untereinander austauschen.

Für die Eisenbahndigitalisierung sind Simulationen von enormer Bedeutung, um die realen Szenarien zu beobachten und die daraus resultierenden Ergebnisse virtuell vorherzusagen. Dies liefert viele wertvolle Einblicke in die Maßnahmen, die im Rahmen des Bahnbetriebs und der Instandhaltung in Zukunft ergriffen werden sollten.

Diese Dissertation zielt darauf ab, die Lücke zwischen Eisenbahnsimulationsressourcen (Modell+Daten) und einer DT-Plattform namens Rail for Future (R4F) zu schließen, indem verschiedene Integrationsstandards, Methoden und Ansätze vorgeschlagen werden, einschließlich Open-Source- oder kommerzieller Softwaretools, Pakete, Bibliotheken, Lizenzen, Schnittstellenstandards, Codebearbeitung, Modell- und Datenverarbeitung. Durch die Verwendung der gesamten Methodik, die in dieser Arbeit vorgestellt wird, können alle Simulationsressourcen, die mit dem Eisenbahninfrastruktursystem zusammenhängen, in die Plattform integriert und verarbeitet werden. Somit können die Eisenbahninfrastrukturmanager und Zugbetreiber in Zukunft die Simulationen, die auf verschiedenen Eisenbahnanwendungsfällen basieren, in der Plattform visualisieren und steuern.

Abstract

Railways have a relatively high share in public and freight transportation, because they provide climate-neutral, money-saving and adventurous mobility in daily life. Aside from their retail in the market, the railway infrastructure operation and maintenance play a significant role in keeping the sustainability of the railway systems, which is highly complex and elaborate. For this, the railway infrastructure is tended to be digitalized, which helps to predictively monitor and maintain different subsystems of the system including the rail vehicle, bridge, tunnel, track and turnout in a virtual environment. Since few years, the digital twin (DT) technology is foreseen as a high potential candidate to enhance the digitalization task for the railway sector. In this case, the real system interacts with its virtual side, where the two parties exchange model and data including simulation between each other.

For the railway digitalization, simulations are enormously important to observe the real world scenarios and then to predict their consequent outputs virtually beforehand. This gives many valuable insights about actions, which should be done inside of the railway operation and maintenance in future.

This dissertation aims to fill the gap between railway simulation assets (model+data) and a DT platform called Rail for Future (R4F) by suggesting to apply various integration standards, methods and approaches including open-source or commercial software tools, packages, libraries, licenses, interface standards, code editing, model and data processing. By using the entire methodology, provided from this work, all of the simulation assets, related to the railway infrastructure system, are able to be integrated and processed in the platform. Thus, the railway infrastructure managers and train operators can visualize and control the simulations, based on different railway use cases, in the platform in future.

Table of Content

1	Introduction	1
1.1	Motivation.....	1
1.2	Problem Description.....	3
1.3	Main Objective	6
1.4	Research Questions.....	6
1.5	The Rail4Future Project Framework	6
1.6	Relevant Publications.....	9
1.7	Research Design and Methodology	10
1.8	Organizational Structure of the Work	18
2	Theoretical Foundations	20
2.1	Digital Twin Concept	20
2.2	Key Technologies.....	22
2.2.1	Interface Standards	22
2.2.2	Open Simulation Platform.....	26
2.2.3	Semantic Web	27
2.2.4	Object-oriented Programming.....	28
2.2.5	DevOps and CI/CD.....	29
3	State of the Art	38
3.1	Literature Research Methodology	38
3.2	Topic Classification	44
3.2.1	Digital Twin for Railway Infrastructure Systems	44
3.2.2	Enabling Simulations in Digital Twins	49
3.3	Research Gaps and Contributions	58
4	Methodology and System Architecture	60
4.1	Task Definition	60
4.1.1	Asset Integration	60
4.1.2	Asset Processing.....	62
4.1.3	Role and Importance for the Rail4Future Project	64
4.2	Asset Integration and Processing Methodology	64
4.3	Automation and Management	72

4.3.1 Main Approach	73
4.3.2 Pipeline Design and Implementation	75
4.3.3 File System Structure in Version Control	79
4.3.4 Documentation of Results	82
4.4 Co-Simulation Implementation	83
5 Validation	87
5.1 Simulation Use Case Examples	87
5.1.1 Residual Life Time (RLT) Calculation of a Railway Steel Bridge	87
5.1.2 Multibody Simulation (MBS) of a Railway Vehicle	90
5.1.3 Anti-Slip Traction and Vehicle Speed Control of an MBS Railway Vehicle	93
5.1.4 ML-based Surrogate Model of an MBS Railway Vehicle	95
5.2 Results and Discussion	98
5.2.1 Use Case Results	98
5.2.2 Discussion	107
6 Conclusion	116
7 Limitations and Future Work	120
8 References	123
Appendix	140
A.1 Description Files	140
A.2 Technical Documentation	154
A.2.1 Functional Mock-up Interface	154
A.2.2 System Structure and Parameterization	156
A.2.3 Distributed Co-Simulation Protocol	159
A.2.4 Open Simulation Platform	161
A.2.5 The R4F Ontology	163
A.2.6 DevOps and CI/CD	165

List of Abbreviations

2D	Two Dimensional
3D	Three Dimensional
5G	The fifth generation of cellular network technology
ACC	Adaptive Cruise Control
ADAS	Advanced Driver-Assistance System
AI	Artificial Intelligence
AIP	Asset Integration and Processing
API	Application Programming Interface
BIM	Building Information Modelling
CAD	Computer Aided Design
CAN	Controller Area Network
CAT	Code Analysis Tool
CI/CD	Continuous Integration / Continuous Delivery or Continuous Deployment
CLI	Command Line Interface
CLT	Command Line Tool
CNN	Convolutional Neural Networks
COMET	Competence Centers for Excellent Technologies
CPS	Cyber Physical System
CPU	Central Processing Unit
CSI	Co-Simulation Implementation
CSV	Comma-Separated Values
DAS	DevOps Automation for Simulations
DCP	Distributed Co-Simulation Protocol
DDS	Data Distribution Service

Dev	Development
DevOps	Development and Operations
DSL	Domain-Specific Programming Language
DSRM	Design Science Research Methodology
DT	Digital Twin
ESDT	Enabling Simulations in Digital Twins
FE	Finite Element
FEM	Finite Element Method
FMI	Functional Mock-up Interface
FMU	Functional Mock-up Unit
GDPR	General Data Protection Regulation
GPL	General-Purpose Programming Language
GUI	Graphical User Interface
HiL	Hardware in the Loop
HTTP	Hypertext Transfer Protocol
I/O	Input / Output
IEEE	Institute of Electrical and Electronics Engineers
IFC	Industry Foundation Classes
IoT	Internet of Things
IP	Intellectual Property
IP Address	Internet Protocol Address
IPSA	Integration and Processing of Simulation Assets
IPSS	Industrial Product-Service Systems
ISA	Instruction Set Architecture
IT	Information Technology
JSON	JavaScript Object Notation
LC Gate	Level Crossing Gate

LSTM	Long Short-Term Memory
m	Meter
MAE	Mean Absolute Error
max	Maximum
MB	Megabyte
MBS	Multibody Simulation
MBSE	Model-Based Systems Engineering
MDPI	Multidisciplinary Digital Publishing Institute
min	Minimum
ML	Machine Learning
N	Newton
NARX	Nonlinear Autoregressive Exogenous Model
nMAE	Normalized Mean Absolute Error
nRMSE	Normalized Mean Squared Error
ODE	Ordinary Differential Equations
OOP	Object Oriented Programming
Ops	Operations
OS	Operating System
OSP	Open Simulation Platform
OSP-IS	Open Simulation Platform – Interface Specification
OWL	Ontology Web Language
ÖBB	Österreichische Bundesbahnen
PC	Personal Computer
PID	Proportional-Integral-Derivative
PLM	Product Lifecycle Management
PNG	Portable Network Graphics
R4F	Rail for Future

Rail4Future	Railways for Future: Resilient Digital Railway Systems to enhance performance
RailML	Railway Markup Language
RAM	Random-Access Memory
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
RDM	Research Design and Methodology
RL	Reinforcement Learning
RLT	Residual Lifetime
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
ROS	Robot Operating System
RQ	Research Question
SCM	Source Code Management
SDTR	Simulations and Digital Twins for Railways
SHM	Structural Health Monitoring
SiL	Software in the Loop
SME	Small and Medium-sized Enterprises
SSB	Structure Signal Dictionaries
SSD	System Structure Description
SSH	Secure Shell
SSM	System Structure Parameter Mappings
SSP	System Structure and Parameterization
SSV	Structure Parameter Values
TCP	Transmission Control Protocol
TU Wien	Technische Universität Wien
UAV	Unmanned Aerial Vehicle
UAS	Unmanned Aircraft System

UDP	User Datagram Protocol
UI	User Interface
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
USB	Universal Serial Bus
UTF-8	8-Bit UCS Transformation Format
VCS	Version Control System
VE	Virtual Entity
ViV	Virtual Vehicle Research GmbH
VM	Virtual Machine
VR	Virtual Reality
VRVis	Vienna Research for Visual Computing
VTI	Vehicle and Track Interaction
WiFi	Wireless Fidelity
XML	Extensible Markup Language
XR	Extended Reality

List of Figures

Figure 1-1: Comparison between the current and envisioned states.	5
Figure 1-2: Landscape of the R4F Platform. [147]	8
Figure 1-3: An Overview of the DSRM Process Model. [102]	11
Figure 1-4: Overview of the Applied Research Design based on the DSRM.	15
Figure 1-5: Research Design and Methodology Process Chart incl. Research Cycles as Further Concretized.	17
Figure 2-1: An overview of the DCP implementation [25].....	25
Figure 2-2: Structure comparison of a VM and container. [146].....	32
Figure 4-1: Model standardization approach. [76].....	69
Figure 4-2: Manual code processing for model integration with FMI. [29]	70
Figure 4-3: Model simulation approach. [76]	72
Figure 4-4: Automation and management approach. [77].....	75
Figure 4-5: An overview of the centralized co-simulation implementation technique based on FMI and SSP.	85
Figure 4-6: An overview of the OSP-based distributed co-simulation technique.	86
Figure 5-1: Bridges used for the RLT calculation. [28, 76]	88
Figure 5-2: MBS railway vehicle modelled in Simpack.....	91
Figure 5-3: Input & output definition and integration of the MBS vehicle with FMI.	92
Figure 5-4: A structural overview of the anti-slip traction and vehicle speed control co-simulation model as containerized using the FMI and SSP standards. [77]	95
Figure 5-5: ML-based surrogate modelling of the Simpack MBS railway vehicle model. [149].....	96
Figure 5-6: An approach to integrate the ML-based surrogate model into the R4F Platform.....	97
Figure 5-7: Life time and damage sum results of the RLT calculation of the dummy bridge delivered from AIT. (Python vs. FMI-based SSP simulation results)	99
Figure 5-8: Life time and damage sum results of the Eschenau bridge from ÖBB Infrastruktur AG. (Python vs. FMU simulation results).....	100

Figure 5-9: Life time and damage sum results of the Mürzbrücke from ÖBB Infrastruktur AG. (Python vs. FMU simulation results).....	101
Figure 5-10: Life time and damage sum results of the Schellhamnergasse bridge from Wiener Linien GmbH. (Python vs. FMU simulation results)	101
Figure 5-11: Vertical deflection results of a primary spring located in a bogie belonging to the MBS vehicle from ViV. (Simpack vs. FMI-based SSP simulation results)	103
Figure 5-12: Actual wheel slip, vehicle speed and wheel speed results of the anti-slip co-simulation model. (SIMAT vs. SSP simulation results)	104
Figure 5-13: Validation results of the ML-based surrogate MBS vehicle model as normalized (predicted vs. real).	106
Figure 5-14: Comparison of simulation results of the anti-slip co-simulation model to the outputs resulting from the OSP-based distributed co-simulation process. (SIMAT vs. OSP vs. SSP).....	111
Figure 5-15: CPU and RAM average resource consumption results. (SSP Centralized vs. OSP Distributed Co-Simulation)	113

List of Tables

Table 3-1: Topic classification for the state-of-the-art investigation.....	39
Table 4-1: Sub-processes of the CI/CD pipeline. [77]	76
Table 4-2: File system structure of the asset application released in the R4F Platform.....	79
Table 4-3: Result documentation of the automated AIP.....	83
Table 5-1: Scenario parameters for the MBS vehicle.....	92
Table 5-2: nRMSE percentage values between the raw model (SIMAT) and FMI-based SSP simulation results.	105
Table 5-3: Resource consumption results (average).....	114

1 Introduction

1.1 Motivation

For several decades now, physical domains (e.g., cars, trains) are tended to be digitalized, because these would then be analyzed, evaluated, monitored, maintained, visualized and validated in a virtual world without utilization of any physical components of the domains. This would help to manage the life cycle of the domains. For example, Kaewunruen and Lian [68] preferred to use the Building Information Modelling (BIM), assisted by the Digital Twin (DT) technology, to manage the life cycle of railway turnouts. They also found out that it significantly improves collaborative work among all stakeholders, who can maintain, visualize and predictively analyze the turnouts in the DT.

Simulations significantly helps to predictively evaluate and validate the dynamical behavior of subsystems, belonging to the physical domains, in real life. Therefore, the simulations are indispensable and have an important part in observing the domains in a digital environment. Since few decades, DTs have been foreseen as a high potential solution to build and then optimize the environment. The DT represents a physical domain such as railways by providing bi-directional data flow between the physical and virtual components of the domain which helps to keep the domain tracked, monitored and maintained in real life. Sepasgozar [122] highlighted the DT over other information systems (e.g., Digital Shadow) because the bi-directional data flow of the DT assists users to collaboratively control and visualize the physical system. An Austrian research project, which is called Rail4Future (Railways for Future - Resilient Digital Railway Systems to Enhance Performance), led by Österreichische Bundesbahnen (ÖBB) with around 20 industry and research partners, and funded by FFG as a COMET (Competence Centers for Excellent Technologies) project, was applied between April 2021 and September 2024 to create and develop a virtual validation framework [108]. This framework allows infrastructure managers and train operators to predictively analyze, monitor and maintain a holistic large-scale railway infrastructure system by using simulations. Zhou et al. [147] conceptualized the Rail for Future (R4F) digital twin platform to provide such an integrated framework. This platform facilitates structured interaction and communication between the infrastructure system

and simulations of its constituent subsystems (e.g., turnout, track), ensuring coherent integration within a unified digital environment. This also helps to reliably, predictively and cost-effectively analyze and evaluate the behavior, condition and measure impacts of the infrastructure system in a digital environment. These simulations consist of various assets (models and data), belonging to different railway use cases (e.g., multibody simulation of a railway vehicle over track), which are to be seamlessly integrated and processed in the platform in order these simulations to work properly. This dissertation aims to fill the research gap between these simulation assets and a railway DT framework by providing asset integration and processing (AIP) solutions, that help to adapt these assets to a DT platform at open-source level such as the R4F Platform in terms of description transparency, platform compatibility, tool independence, file portability and asset quality control. As a use case example, this thesis supplies the integration and processing of a multibody simulation (MBS) model, representing the ride of a virtual railway vehicle over track, in the DT platform. Besides, this work proves the integration and processing of a simulation asset, containing an algorithm to calculate the residual life time (RLT) of a railway steel bridge, in the platform. Moreover, this work shows that an ML-based simulation asset, enabling large-scale simulations over several kilometers of track, can be integrated and processed in the DT platform. Furthermore, this thesis provides useful techniques to implement co-simulations into the platform, where multiple simulation models are interconnected to offer functional enhancements within the corresponding use case. These can be either centrally performed in one computer, achieved for the Rail4Future project, or their execution can be distributed into different computer machines as an alternative solution, which is expected to strategically reduce the computational resource consumption and protect the intellectual property of the simulation models. In addition to supplying AIP and co-simulation implementation solutions, this thesis provides a proposal methodology to increase the automation level of the entire AIP in the platform, and to easily manage it by using key open-source software technologies, which are relevant to the Development and Operations (DevOps) context and continuous integration / continuous delivery (CI/CD) process. This is then expected to reduce the AIP time and money effort, improve the asset quality, version control and help to effectively manage the automation process based on a workflow in a collaborative work environment. To apply the entire asset integration, processing, its automation and management methodology to the DT

platform and also to implement co-simulation frameworks in a centralized and distributed fashion, some useful key technologies are picked, which are therefore mentioned in this work.

Even if boundaries (e.g. tool-dependency) come out while integrating simulation systems into a DT platform, integration solutions can be found and then developed by time. In this work, open-source interface standards such as Functional Mock-up Interface (FMI), System Structure and Parameterization (SSP) and Distributed Co-Simulation Protocol (DCP) are likely used because they provide significant advantages to seamlessly integrate the railway assets in the DT platform. Besides, the object-oriented programming (OOP) is demonstrated to automate the integration process. In addition, a demo virtual machine (VM) of the R4F Platform is built and different software tools, packages and libraries are installed in it to verify the functionality of these simulation assets in the real R4F Platform.

In this section, we introduce this dissertation by motivationally defining and describing problems and main/sub objectives. Besides, we mention research questions (RQ), which are derived from the objectives and then refined for this work. Moreover, the Rail4Future project framework and the relevance of this thesis to this framework are shown and described. Furthermore, we briefly show five publications, that are relevant to this work and have already been accepted for conferences, thus officially published, how the research process is designed, what kind of research methodology is followed to create this dissertation project, and how this thesis is structurally organized.

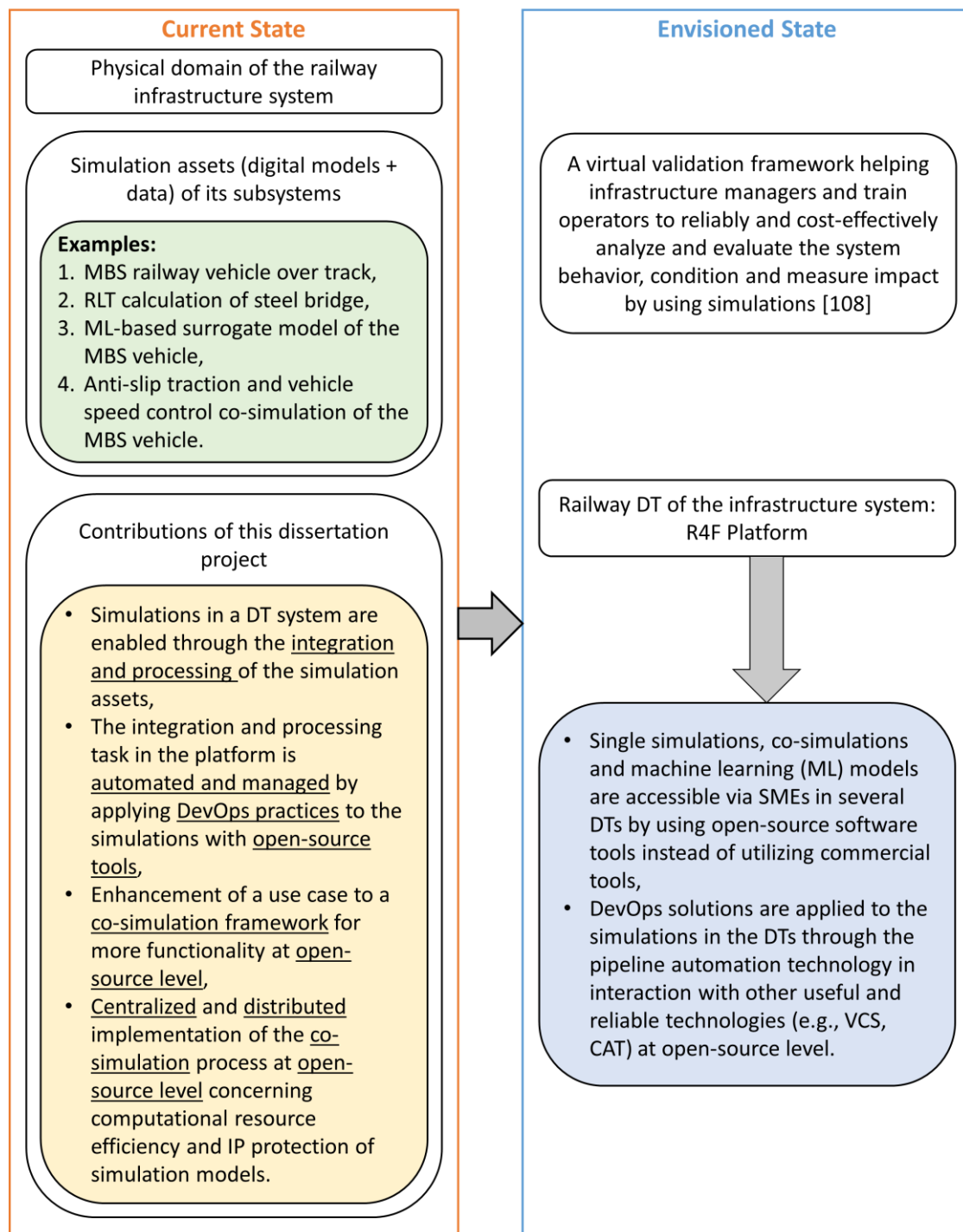
1.2 Problem Description

Beside designing the virtual platform for the digitalization of the railway infrastructure system, it is also important to get comprehensive insights into the principal functionality of various railway simulation assets to seamlessly integrate and interoperate them with each other in the platform. These assets possess a highly complex system structure beginning from the data sets ending with calculation algorithms, which take relatively much time to be learnt and then adapted to the R4F Platform. Besides, these assets are tool-dependent, lack of descriptions and platform compatibility. Furthermore, there are some simulation assets, which need to communicate each other properly (uninterrupted, real-time oriented) in terms of co-simulation. In addition, the asset simulation processes, software licensing, installations and configurations

to complete and easily manage the AIP task require extensive know-how, can take relatively much time, money and effort. This can be handled by automating and managing the AIP as demonstrated in this dissertation. Therefore, some available key technologies, approaches and software for the integration and processing of the railway simulation assets, its automation, management, and co-simulation implementation in the R4F Platform are named in this work.

Based on the above problem description, Figure 1-1 provides an overview of comparison between the current state and envisioned state, planned to be achieved for the Rail4Future project with help of this dissertation work.

Figure 1-1: Comparison between the current and envisioned states.



1.3 Main Objective

First, this dissertation aims to give comprehensive insights into the design and development of a methodology to seamlessly integrate and process any kind of railway simulation asset in a railway DT framework at open-source level such as the R4F Platform. Second, it purposes to enable large-scale simulations of the asset in the platform by applying the proposed methodology, which is important to get insights into the holistic large-scale railway infrastructure system in a virtual environment. Surely, the large-scale simulations take much time, which can be significantly reduced by using ML-based artificial intelligence (AI) approaches (from several minutes to few seconds for plenty of kilometers track), which is already handled in Zhou et al. [149]. Besides, this work aims to provide valuable insights into the implementation of co-simulations into the platform, where multiple simulation models can interact with each other to imagine a variety of possible scenarios (e.g., accelerating / deaccelerating railway vehicle). In addition, this work strives for automation and management of the AIP task by applying DevOps practices to the simulations through the use of CI/CD tools in the platform at open-source level. Based on the problem description and state of the art, this work has the main objective:

- Integration of various simulation models & data of several levels into a holistic simulation and virtual validation framework.

1.4 Research Questions

The following RQs are covered in this dissertation:

- RQ1. What methodologies and key technologies can be applied to realize the asset integration and processing task at open-source level?
- RQ2. How can the asset integration and processing task be automated and managed at open-source level? What challenges and limits are expected to be encountered while striving for full automation?
- RQ3. How can co-simulations be distributed into different machines without sharing the simulation models due to intellectual property (IP) protection of the models?

1.5 The Rail4Future Project Framework

The Austrian Rail4Future project, which took place between April 2021 and September 2024 under the leadership of ÖBB, and funded by FFG as a COMET

project as previously mentioned, covers three main areas. These are as follows: [108]

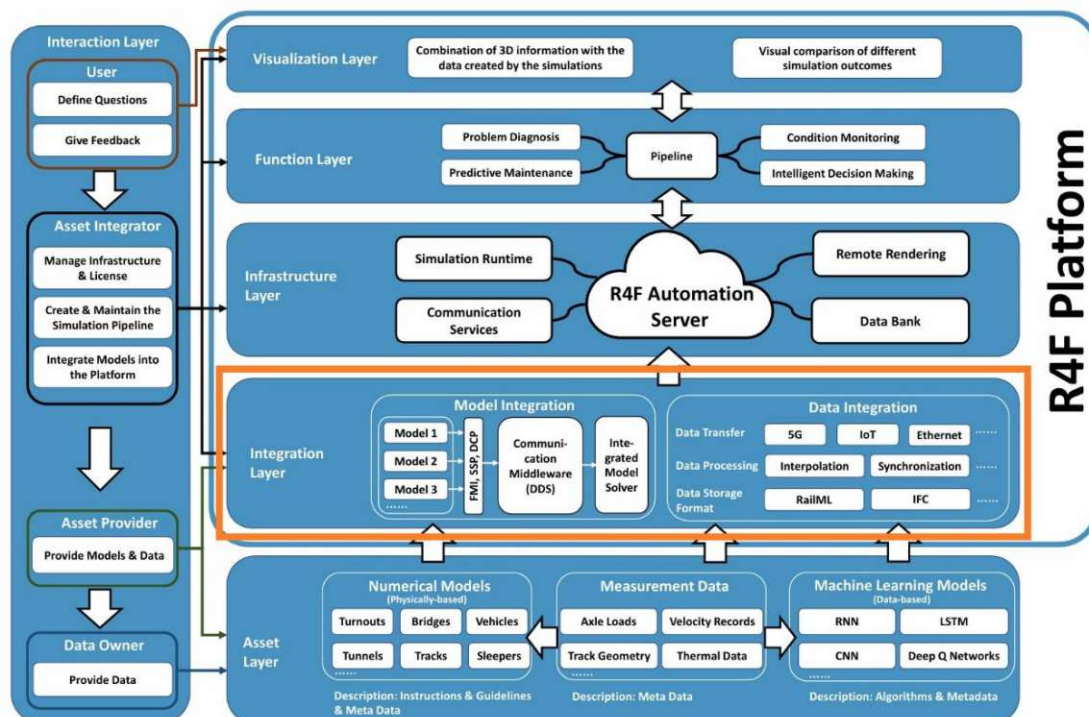
- Area 1 - Simulation Platform for Railway Systems: This area aims to integrate knowledge, model and data, which come from both the other areas, into the virtual validation and simulation framework. Besides, this area feeds these two areas with information, coming from simulation results and predictive analysis, in a closed-loop manner.
- Area 2 - Reliable Railway Tracks: It supplies a sound knowledge about the system interaction and the underlying causes of railway tracks' behavior and its effect on rail vehicles.
- Area 3 - Reliable of Bridges and Tunnels: This area provides a theoretical foundation and understanding of predictive structural analysis and evaluation of railway structures such as steel bridges and tunnels.

Area 1 is the main area, where the virtual validation and simulation framework is conceptualized as the DT-based R4F Platform by Zhou et al. [147]. In terms of simulation asset integration, this dissertation work's contribution to this platform in this area is addressed below, which is based on the landscape of this platform shown in Figure 1-2.

This doctor's thesis mainly focuses on the Integration Layer, which is the bottom part of the R4F Platform (see Figure 1-2 [147], orange marked). In this layer, all the simulation models and data are imported from the Asset Layer and then integrated in the platform. Before that, Asset Provider from the Interaction Layer provides these models into the Asset Layer, where all assets incl. machine learning (ML) models, measurement data and numerical models of different railway subsystems are collected as raw. Besides, some of these models, included in the simulation assets, need to be cooperatively simulated with each other, namely co-simulated to gain more functionality from these. For this, co-simulations are integrated into the platform, occurring on the Integration Layer, which this dissertation work shows and demonstrates as previously mentioned. After the integration process occurring in the Integration Layer, all the integrated assets are further processed in an R4F Automation Server in the Infrastructure Layer. In this layer, simulations are performed through data management in data bank, communication services for interaction of the simulations with the users, and remote rendering technology helping to properly visualize the assets in the platform. The entire automation workflow of the simulation process is defined in simulation pipelines implemented in the Function Layer. In this layer, various functionalities such as condition monitoring, intelligent decision making,

problem diagnosis and predictive maintenance are provided to the pipelines, by which the simulations help the users incl. the railway infrastructure managers and train operators to maintain and monitor their holistic large-scale railway infrastructure resiliently. Lastly, the Visualization Layer improves the visual representation of all the assets for the users. The user control of these assets is assured through user interfaces (UI) in this layer as well.

Figure 1-2: Landscape of the R4F Platform. [147]



In accordance with the R4F landscape proposed by Zhou et al. [147] in Figure 1-2, this dissertation acts as an Asset Integrator, who is responsible for managing and organizing simulation pipelines, located in the Function Layer of the R4F Platform, incl. software licenses in accordance with feedbacks obtained from users. These pipelines help to manage and execute an entire automation workflow, assisting to continuously integrate and process simulation assets of different railway use cases. By this, a simulation asset application is built, tested, delivered and finally deployed into the Visualization Layer of the platform for the users. Depending on use cases, the Asset Integrator needs to get insights into various open-source and/or commercial software tools, which run simulations of raw models belonging to the assets, have configurations, functions and licenses to get to know deeply. After comprehending the main principle of the simulation process and its interaction with inputs and outputs,

they can make the simulation fit to the platform by applying some potential key technologies focusing on the asset integration. In this work, first, open-source interface standards, provided from Modelica Association, (e.g., FMI, SSP) are found useful, practical, cost-saving and reliable for the asset integration task, which is defined in [Subsection 4.1.1](#). Besides, a virtual test environment, having the same compatibility as the R4F Platform, needs to be created and developed, where the Asset Integrator can experimentally integrate and process these assets to verify their integrity, suitability and fidelity for the users of the platform. Regarding to the design and development of this environment, it is also important to notice that on the one hand, open-source software tools are utilized to stay at open-source level, on the other hand, self-hosted hardware, software and data are always preferred instead of externally-hosted ones (e.g., web-based cloud services) due to data protection reasons. In addition, the simulation pipelines, previously mentioned, can automatically integrate, process these assets in an automation server, which might be the R4F Automation Server in the Infrastructure Layer of the platform, in conjunction with other key technologies (e.g., databases, version control) by providing a manageable automation workflow, and can be built by using 3rd party open-source software tools (for more details see [Subsection 2.2.5](#)). Therefore, the use of the pipeline technology in the test environment is a key fact to demonstrate the automation and management of the AIP task at open-source level for the research of this dissertation work, where different approaches, software tools and libraries for the automation and management task are proposed to use (see Kugu et al. [77]).

1.6 Relevant Publications

As part of this doctoral thesis, two publications were personally presented to a broad audience at international conferences. There are three more publications, which are also relevant to this dissertation because of overlapping topics mentioned in this work.

- 1) **Kugu, O.**, Zhou, S., Nowak, R., Müller, G., Reiterer, S.H., Meierhofer, A., Wurth, L. & Grafinger, M. (2023, December), An FMI- and SSP-based Model Integration Methodology for a Digital Twin Platform of a Holistic Railway Infrastructure System. In Modelica Conferences (pp. 717-726). URL: <https://doi.org/10.3384/ecp204717>
- 2) **Kugu, O.**, Zhou, S., Reiterer, S.H., Schwaiger, M., Wurth, L. and Grafinger, M. (2024), Pipeline-based Automated Integration and

Delivery Testing of Simulation Assets with FMI/SSP in a Railway Digital Twin. American Modelica Conference 2024. URL:

<https://doi.org/10.3384/ecp207189>

- 3) Zhou, S., **Kugu, O.**, Reiterer, S.H., Wurth, L. & Grafinger, M. (2024), A Reinforcement-Learning-based Parameter Tuning Methodology for Traction Control in the Holistic Railway Digital Twin System. CIRP Design 2024 (Vol. 128, pp. 828-833). URL: <https://doi.org/10.1016/j.procir.2024.06.040>
- 4) Zhou, S., Dumss, S., Nowak, R., Riegler, R., **Kugu, O.**, Krammer, M. & Grafinger, M. (2022), A Conceptual Model-based Digital Twin Platform for Holistic Large-scale Railway Infrastructure Systems. CIRP Design 2022 (Vol. 109, pp. 362-367). URL: <https://doi.org/10.1016/j.procir.2022.05.263>
- 5) Zhou, S., Meierhofer, A., **Kugu, O.**, Xia, Y. & Grafinger, M. (2023), A Machine-Learning-based Surrogate Modeling Methodology for Submodel Integration in the Holistic Railway Digital Twin Platform. CIRP Design 2023 (Vol. 119, pp. 345-350). URL: <https://doi.org/10.1016/j.procir.2023.02.141>

1.7 Research Design and Methodology

This subsection aims to present how the research to successfully finish this dissertation work is designed, what tasks for the research are performed, and what methods are followed and applied to perform these tasks. In order to clearly show and describe the entire research design and methodology through a process chart, the design science research methodology (DSRM) approach, proposed by Peffers et al. [102], is preferred to be applied in this dissertation work. The reason for this is that the DSRM is considered for the information systems area, which work at the intersection between information systems (IT) technology and organizations like this work [102]. The area belongs to the type of an applied research, which is based on solving problems concerning a society or an industrial/business organization [102, 73]. As clearly realized, the research for this dissertation is also an applied research, because this work proposes various comprehensive and insightful AIP, automation & management, and co-simulation implementation solutions to solve problems mentioned in [Subsection 1.2](#), which are practice-oriented, related to IT and concern a railway infrastructure organization. The organization can then predictively analyze, evaluate and conditionally monitor the railway infrastructure system by using the simulations in a DT platform. Deriving from

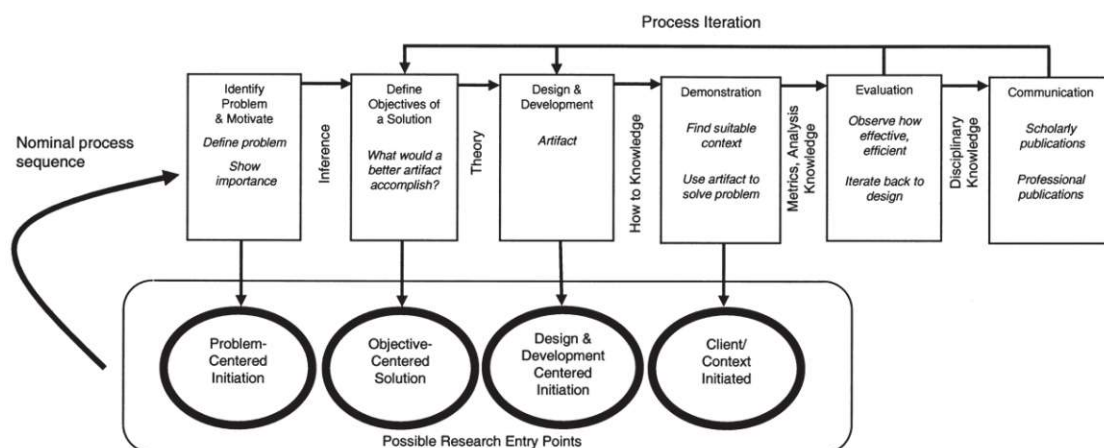
the problems, the main objective of this work (see [Subsection 1.3](#)) focuses on the design and development of an IT-related methodology making the DSRM approach highly preferable for the research of this work.

Figure 1-3 gives an overview of the DSRM process model, which contains six activities performed step by step:

1- *Identify Problem & Motivate*: This activity aims to describe the research problem, and then encourage the researcher and audience to create and develop an artifact, which provides a solution to the problem. Appreciating the value of the solution helps them to understand the entire reasoning and background of the research. Information about the state of the problem, and the importance of the solution is caught as resources from this activity.

2- *Define Objectives of a Solution*: This activity provides insights into objectives of a solution, which rationally derive from the problem description and applicable options. These objectives reveal what is quantitatively or qualitatively achieved after solving a problem by using the artifact. For this activity, information about the state of the problem, available solutions and their impact is required.

Figure 1-3: An Overview of the DSRM Process Model. [102]



3- *Design & Development*: This activity is the main research focus, where the research artifact (e.g., methods, approaches, models...) is created and developed under consideration of its functionality and architecture. Knowledge of theory is needed to reach this activity from the *Define Objectives of a Solution* activity.

4- *Demonstration*: It is the activity, where the artifact is applied to practical uses (e.g., experiments, simulations, case studies...). This activity aims to prove the applicability of the artifact to solve a problem and thus to achieve the objectives. A clear instruction about the use of the artifact to solve the problem is a necessary resource to do the demonstration activity.

5- *Evaluation*: In order to show the validity of a problem solution, provided by the artifact, this activity is considered. For this, information about relevant metrics and analysis techniques is required. The validity can be achieved in various ways such as analytical result comparison between measurements and simulations, gaining outputs from user feedback, surveys or case studies as any comprehensible experimental or logical evidences. After finishing the evaluation activity, the researcher can determine if they need to head back to the *Design & Development* activity or go on to the *Communication* activity by completing and writing down the research with further improvements.

6- *Communication*: It simply purposes to communicate the entire research incl. the problem, the significance of it and its solution, the artifact, design, novelty and strength of the artifact, and the artifact's main efficacy to other researchers and audiences. This activity requires disciplinary knowledge, by which the researcher can structure the research process (e.g., systematic literature review, data analysis and validation, conclusion...), and finally publishes it in a written form (e.g., conference papers, dissertation...).

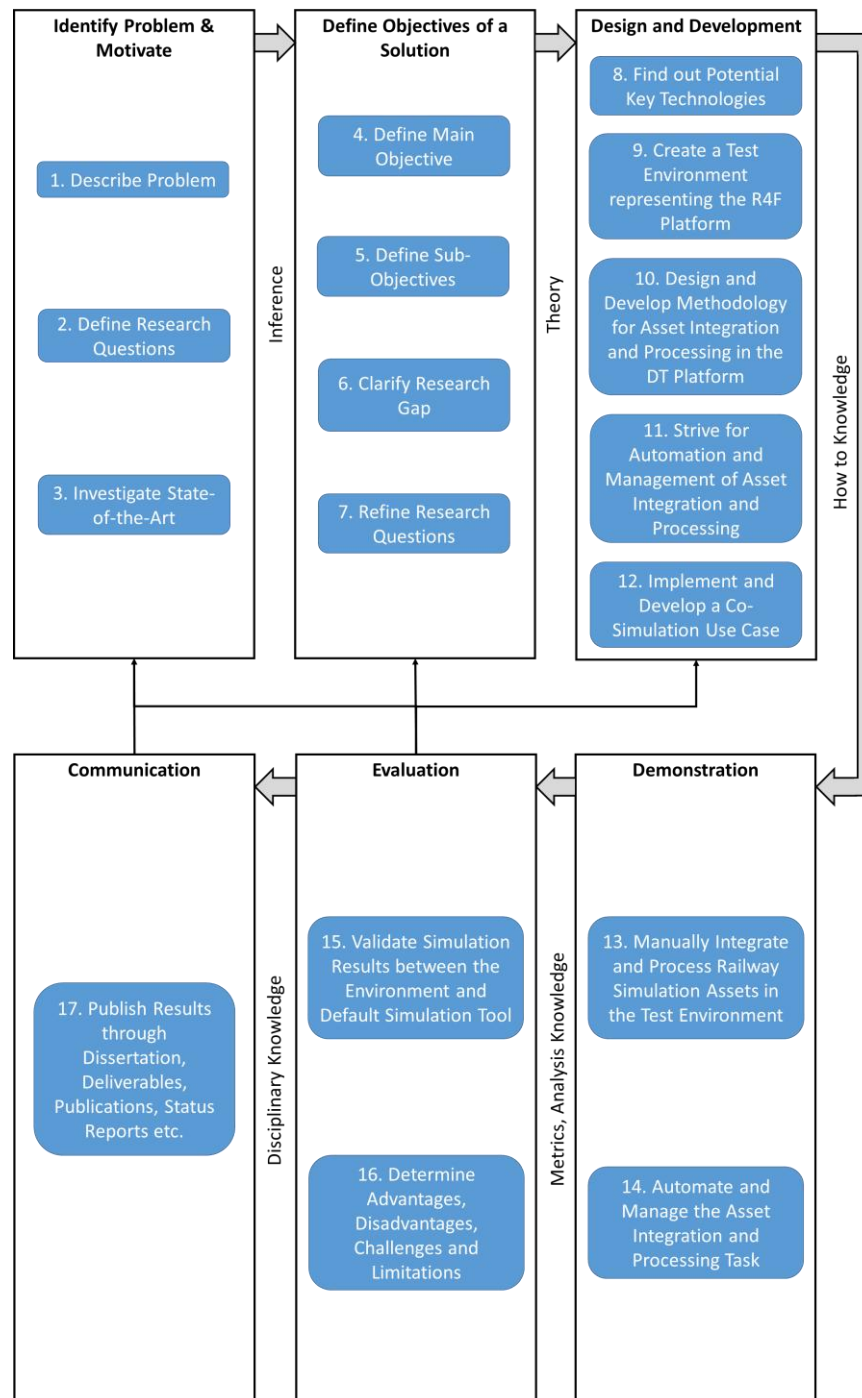
In addition to these six activities, Peffers et al. [102] pointed out four research entry points in their DSRM process model, which are problem-centered initiation, objective-centered initiation, design & development-centered initiation and client/context-centered initiation. These indicate where the research is initiated depending on the case. They also described the demonstration of all these four entry points by mentioning four case studies in their work, which exactly correspond to these four entry points.

Figure 1-4 exactly shows and describes how the research, performed for this dissertation work, is applied to the DSRM process model. First, it is worthy to mention that this dissertation associates with the objective-centered initiation entry point, because the research to design and develop the entire AIP, its automation & management, co-simulation implementation methodology as an artifact is initiated from the main objective (see [Subsection 1.3](#)). This objective comes from the Rail4Future project as a task to make simulations available in

the R4F Platform for a railway infrastructure organization, as comprehended from the name of this work. Then, the six activities are sequentially performed within the research process incl. retrogressive iterations. In *Identify Problem & Motivate* activity, the main problem about the lack of simulations in the platform, and three RQs regarding to the AIP task, its automation & management and co-simulation implementation are defined and described (see [Subsection 1.2](#) and [Subsection 1.4](#)). Besides, the state of the art is investigated to find out potential key technologies, which are helpful for the research process (see [Section 3](#)). Then, in *Define Objectives of a Solution* activity, the main objective and its sub-objectives, which are accomplished by using an artifact in the research process, are defined as mentioned in [Subsection 1.3](#). The research gap is also clarified (see the beginning of this section) and the RQs are refined. This then leads to create and develop the artifact, consisting of the previously mentioned methodology and system architecture, to solve the problem within the *Design and Development* activity (see [Section 4](#)). For this, useful key technologies are found out (see [Subsection 2.2](#)), a test environment is prepared as a representative simplistic and compatible version of the R4F Platform, the AIP task is defined, methods and approaches are designed and developed to perform the task at open-source level, to strive for its automation and management by using open-source software tools and thus applying DevOps practices to the simulations in the environment, and finally to implement co-simulations into the environment in a centralized and distributed fashion are found out for more functionality and IP protection of simulation models. After that, in the *Demonstration* activity, the artifact, namely the entire methodology is demonstrated by manually integrating simulation assets of various railway use cases, mentioned in [Subsection 5.1](#), and then automating and managing the AIP through DevOps practices in the environment to prove the practicability of the methodology for solving the main problem, namely for enabling the simulations in the R4F Platform. In the *Evaluation* activity, first, the simulation results between the environment and default simulation tools are compared to judge the consistency, stability of the simulation processes, and also the reliability of the system architecture and representative environment. Then, overcoming challenges, limitations, typical advantages and disadvantages while demonstrating the artifact in the environment are determined and discussed to raise awareness of the design, development and demonstration of the artifact. In case of the need for further improvements in the research, it is decided to iterate back from the *Evaluation* activity to *Design and Development*,

Define Objectives of a Solution or Identify Problem & Motivate activities. For example, a simulation result is not comprehensible, incomplete (new results are needed) or not showing any reasonable behavior regarding to the simulation process. To improve it, a new RQ is defined, state of the art further investigated, a new sub-objective defined, the artifact is further analyzed and developed. At the final stage, namely in the *Communication* activity, the research is communicated through five publications (incl. attending conferences for paper presentation, see [Subsection 1.6](#)), Rail4Future reports (incl. status reports, deliverables [107]) and this dissertation work after clearly structuring the research process. The option for the retrogressive iteration also applies to the final activity. As a concrete example, a publication is to be further reviewed by internal and external competent reviewers to improve the work before presenting and publishing it to an audience. For this, a new problem and RQ are defined, state of the art extensively investigated, a new sub-objective defined, the research gap clarified due to it, or the entire artifact further revised and improved in terms of reliability, traceability and comprehensibility. As a result of the iterative progress, the publication is effectively finalized.

Figure 1-4: Overview of the Applied Research Design based on the DSRM.



The idea of the refined description of the DSRM model as research cycles with applied methods is inspired from the Zigart's dissertation work [150]. Figure 1-5 indicates the demonstration of it in three research cycles, performed for the research of this dissertation work, incl. applied methods:

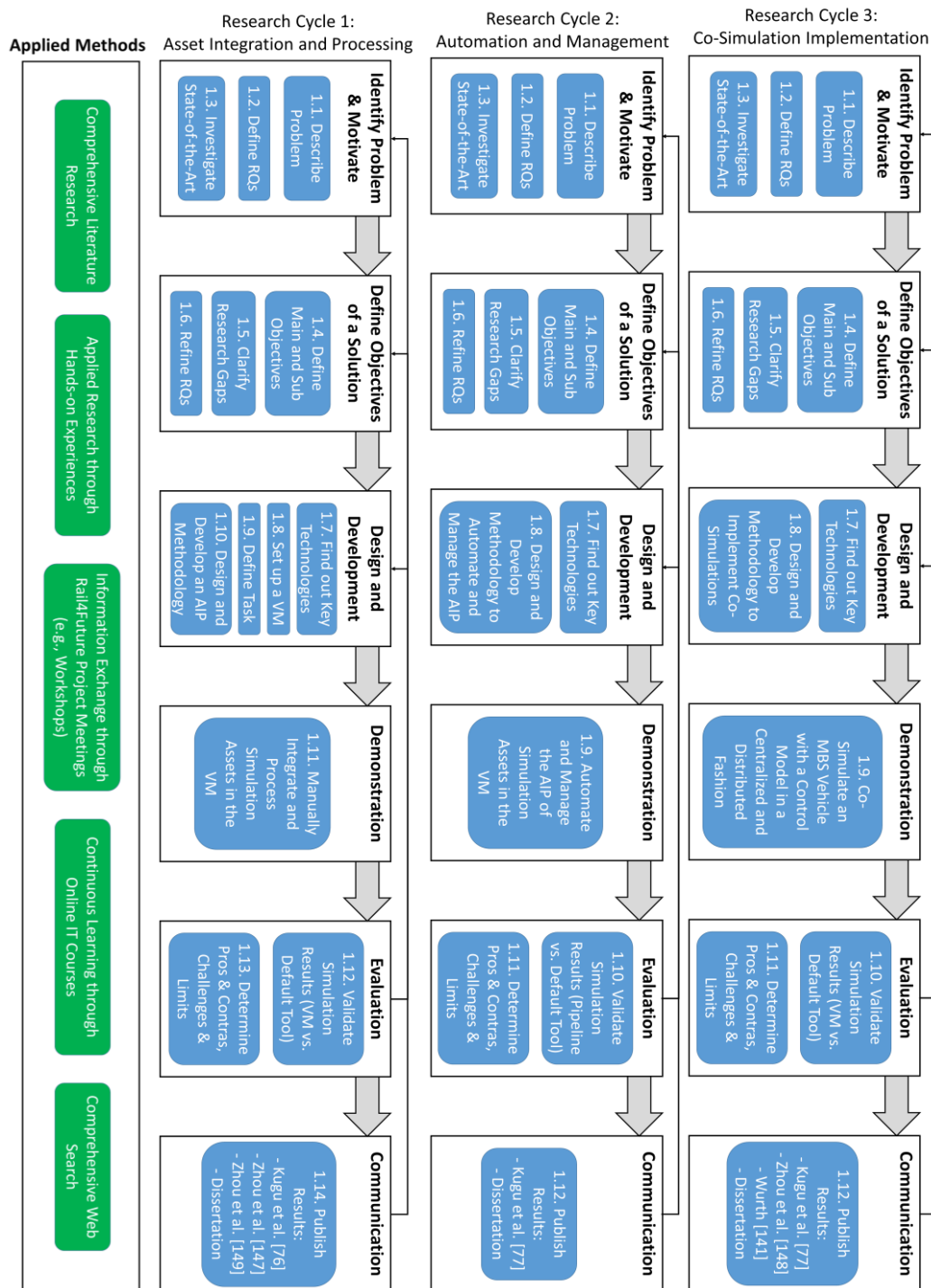
- **Research Cycle 1: Asset Integration and Processing** → This research cycle aims to find out a solution to fill the research gap between the railway simulation assets and R4F Platform by providing an AIP methodology as an artifact.
- **Research Cycle 2: Automation and Management** → This research cycle purposes to reach an optimal solution to automate and manage the AIP task for more resource-, time- and cost-efficiency by applying DevOps practices to simulation asset applications in the R4F Platform as an artifact.
- **Research Cycle 3: Co-Simulation Implementation** → This research cycle helps to enhance simulations as a co-simulation framework, then to implement this into the platform in a centralized way for more functionality by providing a centralized co-simulation implementation approach as an artifact. Besides, this research cycle offers an alternative implementation technique as another artifact, by which co-simulations can be distributed into different machines, installed and configured in the test environment of the research, in order to investigate the computational resource efficiency and IP protection of simulation models in the environment.

For all these research cycles, the methods are applied to the research of this dissertation work:

- **Comprehensive Literature Research:** Relevant literature examples (e.g., conference papers, journal papers...) are found out and then written down to support this thesis.
- **Applied Research through Hands-on Experiences:** The research of this work is put into practice through acquired IT competencies, which was especially important for the *Design & Development* and *Demonstration* activities (e.g., DevOps engineering for automation of the AIP task).
- **Information Exchange through Rail4Future Project Meetings (e.g., Workshops):** This method extremely helps to detect potential key technologies and gain useful insights into requests and use case examples (see [Subsection 5.1](#)), coming from internal and external colleagues, by attending Rail4Future project interviews, which definitely assists to complete the research of this work.
- **Continuous Learning through Online IT Courses:** By this method, the theoretical and practical IT knowledge is expanded by steadily learning IT topics, relevant to the research of this dissertation (e.g., Python programming), to broaden the research perspective while finishing this dissertation project.

- Comprehensive Web Search:** This is also very helpful to easily and quickly find what needs to be known for the completion of the research process.

Figure 1-5: Research Design and Methodology Process Chart incl. Research Cycles as Further Concretized.



1.8 Organizational Structure of the Work

Section 2 introduces the theoretical background of this dissertation work. First, this section presents the core definition and description of the term DT. Then, key technologies, including open-source interface standards, the Open Simulation Platform (OSP) software application, object-oriented programming, semantic web, DevOps- and CI/CD-related software technologies, are listed and described in this section. These are considered and used to design and develop a test environment, representing a simplistic and compatible version of the R4F Platform, to integrate and process railway simulation assets, to automate and manage it there, as well as to implement co-simulations in a centralized and distributed fashion into this environment.

Section 3 provides the state of the art of this dissertation work, showing and describing what literature research methodology is followed, into which subtopics the main topic of this work is rationally divided, which research gaps are filled, and what scientific contributions and novelty are brought by performing the research for this dissertation project as a result of the comprehensive literature research.

Section 4 exposes the entire methodology and system architecture proposed with a workflow-based pipeline in this work. First, this section gives an overview of the AIP task, which is previously mentioned and the part of which in the R4F Platform is clearly addressed by Kugu et al. [77]. It exposes how the AIP task is defined and described in details. This task is proposed as an approach, that is commonly followed in this dissertation to adapt various railway simulation assets to the platform. Based on this, the role and importance of this task for the Rail4Future project is mentioned in this section. Then, the section illustrates what kind of methodology and system architecture are designed, developed, followed, and how that is used with all these key technologies together in a proper way to do the AIP task, then automate and manage it. Finally, this section presents how co-simulations are centrally implemented into the platform, and distributed these into different machines in the test environment of the platform.

Section 5 generally shows the entire evaluation of the workflow-based pipeline and system architecture of the test environment, which are designed and developed through the approaches and methods (see [Section 4](#)), by applying the railway use case examples (see [Subsection 5.1](#)) to the environment. First, this section lists and describes all these use case examples, which are

successfully demonstrated by integrating and processing their simulation assets incl. co-simulations in the test environment, representing a simplistic and compatible version of the R4F Platform. Then, the section presents use case specific validation results, which show the clear comparison between the FMI-based SSP and raw model simulations and thus used for evaluation of the pipeline and system architecture. This would then contribute to the predictive maintenance and conditional monitoring of several railway subsystems by analyzing and further comparing these results with measurements, coming from the physical subsystems, in a railway DT such as the R4F Platform in future. Lastly, this section lists and describes discussion matters, which include benefits of the proposed methodology, challenges and limitations, encountered during the AIP, its automation & management and co-simulation implementation, and comparison between the centralized and distributed co-simulation implementation types.

Section 6 refers to the conclusion and concrete answers of the three RQs, mentioned in [Subsection 1.4](#), in this work. Section 7 explains limitations and future work belonging to this thesis. Lastly, references are listed in Section 8, and example system/model description and parameter files, used for the AIP in the R4F Platform, as well as the extensive description of some key technologies are presented in the Appendix section.

2 Theoretical Foundations

In this section, first, the term DT is described, its application areas are shown, its importance and its differences from other information systems such as physical domain, digital thread, Digital Model and Digital Shadow are clarified. Then, this section exposes the theoretical background of potential key technologies, which play a significant role on the *Design and Development* activity of the research process (see [Subsection 1.7](#)) in this dissertation project.

2.1 Digital Twin Concept

Since 1960s, the DT concept has become famous and is applied to NASA's Apollo program for the first time, where a space vehicle on earth is presented as the twin of the vehicle existing on the space [35]. The twin vehicle represents the operational behavior of the vehicle on space, and was implemented on hardware. Then, the DT term is used by Hernández and Hernández [62] in 1997 for the first time after they modeled a geometrical illustration of an urban environment, consisting of 3D digital models (e.g. of a bridge), and called it DT. As understood from the first concrete example from NASA, this term comes from the twins, who are identical to each other as brothers or sisters. The word "Digital" identifies the version of the twins meaning the virtual representation of a real world object. In this case, there are three objects, two of which are the twins, namely a physical domain, the other one represents the software-based virtual version of the physical one, and the last one of which is the connection between these two domains. In other words, the DT is observed as a contextualized software model, by which the behavior of the physical domain can be imitated in software [24]. The physical domain can be a car, spaceship, plane or train system as typical examples. Digital Model is the virtual version of the real domain, however there is no automatic information exchange between the physical and virtual counterparts in Digital Model like in DT [3]. Of course, the definition of the DT is a matter of discussion regarding to different organizations and communities [24]. As a concrete example of this, Semeraro et al. [121] found out several DT-related publications, then extracted and listed different DT definitions from these in their systematic literature review about the DT technology and its applications. On the other hand, the only difference between DT and Digital Shadow lies on the direction of the automatic information exchange: unidirectional in Digital Shadow (physical object →

virtual object); bidirectional in DT (physical object ↔ virtual object, reversal exchange) [3]. In digital thread, all digital objects are interconnected throughout product life cycle phases helping to track the objects eventually [3].

The DT technology is applied to several application areas, which researchers comprehensively covered and reviewed in their scientific research works. For example, Enders & Hoßbach [35] pointed out the existence of DT applications across various industry sectors such as manufacturing, aerospace, energy, automotive, marine, petroleum, agriculture, healthcare, public sector and mining in their systematic literature review. Semeraro et al. [121] mentioned city management, maritime and shipping sectors additionally after their findings about designing and building DTs for their comprehensive literature research. Besides, Botín-Sanabria et al. [18] called education, construction, business, natural disaster detection, communication, security, logistics, robotics, design and products in their DT review study. Ghaboura et al. [44] also draw the attention to a huge trend of the DT technology for the railways domain while gathering over 100 publications related to the DT applications for this domain, and then choosing 80 of these due to specific criteria (e.g., topic relevance, publication year...) in their comprehensive literature review. As they found out, the DT might be used for the management and maintenance of a railway infrastructure system.

Relevance for this work: By using simulations in the DT technology, static, dynamical behavior and visual insights (e.g., FEM, CAD model) of a physical system can be predictively analyzed, and parameter studies can be performed while comparing and validating the results of the analysis based on the simulations of the system (e.g., operational condition monitoring and predictive maintenance of crossing panels based on the MBS model of a vehicle-track system [86]). Therefore, the users of the DT can be satisfied and complied with the resilient, available and insightful maintenance and monitoring methodology. The physical system might be a railway infrastructure system, which is visualized, predictively analyzed, monitored and maintained by using simulations in the DT of the system as previously demonstrated for the Rail4Future project [108]. For this dissertation, a railway infrastructure system, belonging to ÖBB, is taken into account as a physical domain because this work improves the integration and processing of simulation assets in a railway DT platform, called R4F and conceptualized by Zhou et al. [147] as part of the

Rail4Future project [108], as previously mentioned in [Subsection 1.1](#) and [Subsection 1.5](#).

2.2 Key Technologies

2.2.1 Interface Standards

2.2.1.1 Functional Mock-up Interface

In 2010, the Functional Mock-up Interface (FMI) standard was first introduced as an open-source interface standard in the ITEA2 Modelisar project, and called Functional Mock-up Interface for Model Exchange 1.0 [39]. This technology is already supported by more than 250 tools [133], meaning that it gains a high popularity in research and industry. This standard works on the basis of a black-box approach, aiming that a simulation model is packaged into a black box, which is compatible with an FMI-supported environment [19]. This black box is a ZIP container, which is used to exchange dynamic simulation models as a standardized simulation unit of the models called Functional Mock-up Unit (FMU). Blochwitz et al. [17] gave an overview of the FMI incl. its distribution, description schema, compatible tools. They pointed out the high potential of this standard for an easy model exchange and co-simulation between various software tools in terms of tool-independency.

The summarized technical description of the general structure and components of the FMI standard and FMU file can be found in [Appendix Subsection A.2.1](#). Besides, there is an example model description file of the FMU in Appendix 1.

Relevance for this work: The railway simulation systems consist of models and data (e.g., geometrical parameters), which together are called simulation assets. In usual, the raw simulation models of these assets run in different software tools, which are either commercial or open-source, depending on the use case. By time, more simulation models come out, which means more tools to install and deal with. These models may also have a relatively high data complexity due to the increased number of files or folders. Besides, these models may not be compatible in the corresponding operation system (OS), where these are supposed to run. In addition, the models may lack of descriptions, which expose essential information about the model in a human readable fashion (e.g., model name, default tool, name of input parameters...). To overcome all these, we preferred to standardize the simulation unit of these

models by using the FMI open-source standard, which is important to integrate the assets into the R4F Platform, and therefore design and develop the AIP methodology at open-source level (see [Subsection 4.2](#)). This also means that this standard is one of these key technologies, answering the RQ1 (see [Subsection 1.4](#)). In short, the FMI standard helps to make these models more tool-independent, platform compatible, description transparent and file portable.

2.2.1.2 System Structure and Parameterization

In 2019, the System Structure and Parameterization (SSP) standard is first introduced as a Modelica Association project, providing an open-source interface standard for a tool-independent description, containerization and interchange of system structures and their parameterization, in the SSP specification document (version 1.0) [128]. This standard technology helps to define and describe a system, which contain one or more models as FMU component(s), its parameterization [129]. All these together can be interchanged between different simulation tools [129]. Since a few years, the interest in the SSP technology is growing, which is already supported by more than 10 tools [61]. Ochel et al. [93] mentioned the use of the SSP standard as an exchange format for compound models in the OMSimulator FMI-based co-simulation tool. Hatledal and Fagerhaug [58] interpreted the SSP as an extended version of the FMI standard, helping to simulate not only single components, as well as systems consisting of these components. Besides, Hällqvist et al. [56] demonstrated the interoperability between different domains based on an industrial use case example, where systems simulation and geometric modelling domains are interoperated incl. automatic parameter exchange between them, by using the FMI and SSP standards.

The summarized technical information about the general structure and components of the SSP standard and its SSP file can be found in [Appendix Subsection A.2.2](#). Besides, there is an example SSP system structure description file in [Appendix Subsection A.1](#) (see Appendix 2).

Relevance for this work: The FMI standard, previously mentioned, indeed provides tool independency, platform compatibility, file portability and model description as a ZIP container, however this standard does not supply a system structure, which can have two or more simulation models as co-simulated, and also its description incl. connections. In some cases, a railway use case is to be enhanced to imagine different scenarios, which can be achieved by

implementing a co-simulation framework coming from the use case. As a concrete example of this, a railway virtual vehicle is co-simulated with a PID-based control model to control the anti-slip traction behavior and speed of the vehicle (see [Subsection 5.1.3](#)). In order to provide the co-simulation framework, all these models with their data (assets) are to be connected to each other within this framework. The default software tools of these models can have some particular ability to co-simulate these models, however these have to be installed in this case causing tool dependency. To overcome the tool dependency, and to provide a descriptive system structure and its parameterization for co-simulations at open-source level, the SSP standard together with the FMI standard is considered as a high potential candidate, which significantly helped to integrate co-simulation assets into the R4F Platform for the Rail4Future project, in this dissertation work. Therefore, this technology has an important part in the design and development of the AIP methodology (see [Subsection 4.2](#)) and the co-simulation implementation technique (see [Subsection 4.4](#)) at open-source level. Therefore, the standard is one of the key technologies helping to answer the RQ1, mentioned in [Subsection 1.4](#).

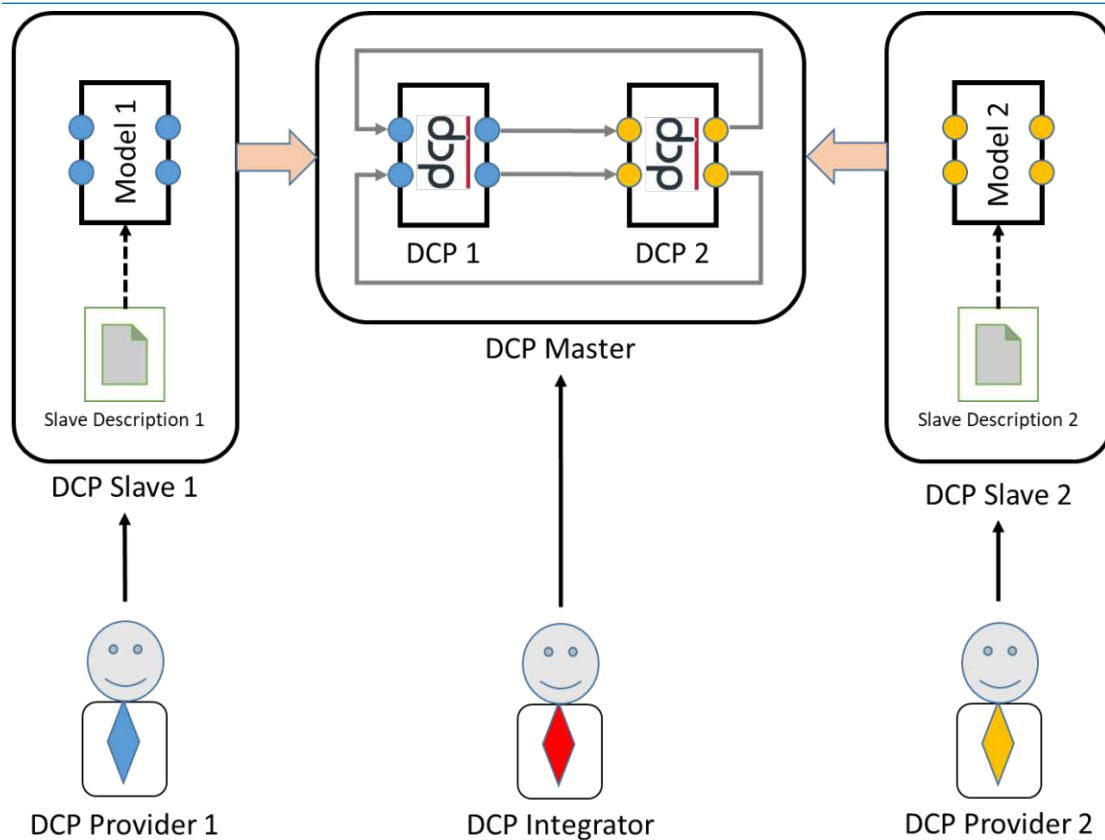
2.2.1.3 Distributed Co-Simulation Protocol

In 2018, the Distributed Co-Simulation Protocol (DCP) standard is first introduced and proposed by Krammer et al. [75] as an open-source interface standard, by using which they aimed to integrate real-time and/or non-real-time systems into simulation environments. This standard follows the master-slave principle, which ensures the integration of one or multiple DCP slaves into a common system (master). In this system, a co-simulation scenario is created. These DCP slaves indicate either a model or a system, which can be real-time and/or non-real-time. [25]

As a concrete example of the use of the DCP standard, Segura et al. [120] implemented this standard into x-in-the-loop simulations, which are used for the verification and validation tests of cyber physical systems (CPS), to perform these in a distributed fashion. They also found this key technology useful and promising to protect the IP of CPS elements (incl. physical model, hardware), and to connect different hardware components to these helping them to achieve higher performance in spite of unexpectedly occurring simulation behaviors, caused by synchronization issues, as a conclusion of their work.

Figure 2-1 indicates an overview of the simplistic DCP implementation approach through the use of the master-slave architecture. The DCP Provider is the legal owner of the DCP slaves, which they provide to the DCP master belonging to the DCP Integrator. Each DCP slave includes a model and an XML-based slave description file (.dcp), which defines typical characteristics of the DCP slave (see Appendix 10). The DCP Integrator is the person, who is responsible for the connection of the DCP slaves to each other to co-simulate these using DCP components in their own virtual platform.

Figure 2-1: An overview of the DCP implementation [25].



The summarized technical information about the DCP standard can be found in [Appendix Subsection A.2.3](#). Besides, there is an example DCP slave description file in [Appendix Subsection A.1](#) (see Appendix 3).

Relevance for this work: In some circumstances, co-simulations need to be performed in a distributed fashion to enhance the IP protection of a simulation model. By this, the model is not shared among different stakeholders anymore, it only stays in the local machine of the model owner. Indeed, there are possibilities to realize this kind of co-simulation approach by using default

simulation tools. However, the co-simulation is tool dependent and the corresponding tools are needed to be installed and started in a virtual platform in this case. In addition, network configuration of these tools incl. IP addresses and ports is an essential topic to take into account for the implementation of this kind of co-simulation into a DT platform. Another important advantage of the distributed co-simulation lies on the computational resource efficiency, which means that the resource consumption (e.g., CPU, RAM) in a particular machine can be reduced by distributing the simulation models into other machines. In addition to that, the distributed co-simulation might help to improve the software license management by providing the software licenses only into the corresponding machines. In these machines simulation models are instantiated, which are designed and developed in default commercial simulation tools requiring these licenses. This might help to save money and energy, thus reducing life-cycle costs. To overcome the tool and communication medium dependency mentioned above, the open-source DCP standard is seen as an alternative solution for a use case implementation into the R4F Platform (see [26]). [Subsection 4.4](#) exactly shows and describes the distributed co-simulation methodology, considered for this dissertation project and indicating how the co-simulation process, an example of which is mentioned in [Subsection 5.1.3](#), might be implemented into the R4F Platform using the DCP standard technology. Besides, this standard might help to answer the RQ3 (see [Subsection 1.4](#)) regarding to the IP protection of the simulation models during the distributed co-simulation process without exchanging these between different stakeholders. According to the DCP implementation overview, shown in Figure 2-1, this dissertation work takes the part of the DCP Integrator, because it works as an Asset Integrator as previously mentioned in [Subsection 1.5](#), and aims to provide the integration and processing of different simulation assets of the holistic railway infrastructure subsystems in the R4F Platform. Therefore, the platform is directly considered as the DCP master and the simulation assets as the DCP slaves for this work. By this, the Asset Provider, mentioned in [Subsection 1.5](#), takes the role of the DCP Provider in this case.

2.2.2 Open Simulation Platform

In 2018, the Open Simulation Platform (OSP) was initiated as the OSP Joint Industry Project (JIP), which is established by DNV GL, Kongsberg Maritime, SINTEF and Norwegian University of Science and Technology NTNU [99]. The

OSP software is open-source and provides a co-simulation environment, based on the FMI and OSP interface specification (OSP-IS), for the maritime industry [97]. The OSP-IS is an extension to the FMI standard providing semantics, which helps to overcome high complexity and difficulty in connection and validation of various simulations, caused by several complex models and systems from different providers [98]. Rindarøy et al. [113] also pointed out that the benefits of the OSP-IS lie on the better comprehensibility and traceability of model interfaces, which are defined by the FMI standard based on variable types and names, by utilizing the semantic technology.

The summarized technical information about the main components of the OSP software can be found in [Appendix Subsection A.2.4](#). Besides, there is an example OSP system structure description file in Appendix 4.

Relevance for this work: Based on above, the OSP software might be a key technology to fill the gap between the distributed co-simulation and DT technology. As an example, Perabo et al. [103] proposed to use the OSP software for their DT modelling and co-simulation tasks, which were performed for maritime industry. In their study, they demonstrated the co-simulation of electric ship power and propulsion systems with controllers as a concrete use case example by using the FMI standard and OSP software. As a result, they found the OSP technology promising to build a DT-based co-simulation environment, which fosters collaboration and works based on standards, for the maritime area while mitigating lifecycle costs and time. This also shows a guiding insight into the use of this key technology, where the railway co-simulation use case example, mentioned in [Subsection 5.1.3](#), might be integrated into the R4F Platform and then performed in a distributed fashion in the platform by using this technology. Therefore, the OSP software might contribute to the design and development of the distributed co-simulation implementation technique (see [Subsection 4.4](#)). Besides, the OSP software application might be one of these key technologies, answering the RQ1 regarding to the integration and processing of co-simulation assets, and the RQ3 regarding to the IP protection of simulation models during the distributed co-simulation process (see [Subsection 1.4](#)).

2.2.3 Semantic Web

As previously mentioned, many different standards and tools are used to integrate and process several railway simulation models and data in the R4F

Platform, which naturally causes high data complexity and redundancy. In order to ease the data exchange and data integration between the users of the platform, the semantic web technology is preferred to be applied to the platform by creating and developing an ontology. This technology is proposed by World Wide Web Consortium (W3C) and relatively popular among researchers and software engineers. For the Rail4Future project, the R4F Ontology is preferred to be used, which is proposed by Kern [72], in order the users to be able to easily reach information (e.g., input & output variables, model metadata) about all these simulation assets, integrated into the platform using the open-source interface standards.

The technical documentation of the R4F Ontology can be found in Appendix [Subsection A.2.5](#).

Relevance for this work: In conclusion, the R4F Ontology is insightful and promising for an easy data exchange and data integration in the R4F Platform. This increases the comprehensibility and traceability of the railway simulation assets, which are integrated and processed in the platform within the Rail4Future project. According to this, the R4F Ontology is found helpful to ease the management of the AIP task, and thus mentioned in this dissertation work. That is why the semantic web technology, which significantly contributed to build the R4F Ontology as found out in Kern [72]'s work, might be an answer to the RQ2 (see [Subsection 1.4](#)).

2.2.4 Object-oriented Programming

Object-oriented programming (OOP) plays a major role on implementing various software applications into different virtual environments. The term OOP comes from the object, which is a group of data and corresponding behaviors [80]. The OOP means writing code directed toward modeling objects, that is used to describe actions of a complex system using the objects [80]. By this, a design idea is applied to a working program, which helps to make the system into work [80].

Relevance for this work: The system, mentioned above might be a simulation system as a concrete example. In the case of this work, the system is defined as an asset application consisting of the railway simulation assets, a simulation code script and a bunch of commands executed to start the code script and thus to run the simulation in the R4F Platform. The code script is a product of the

OOP technology, uses a software tool, different software libraries, packages and functions, possessing objects with data and behavior from the OOP, to do some particular tasks making the asset application useful for the users of the platform. This makes this technology a highly potential candidate for the integration and processing of different simulation assets as asset applications in the R4F Platform. Thus, this technology might be an answer to RQ1 regarding to the realization of the AIP task (see [Subsection 1.4](#) and for more information about the proposed AIP methodology see [Subsection 4.2](#)). By the way, it is worthy to mention that there are different OOP-based languages in the world, which were designed and written based on needs. Python is one of the most popular one among them and was developed by Guido van Rossum, when it first appeared in 1991 as Python 0.9.0. The Python OOP language is used to build the asset applications for this research work, because this language is easy-to-code, platform independent, dynamically typed, open-source and high-level. Besides, Python enables proper interaction and communication between the railway simulation assets, adapted by using the interface standards, and the R4F Platform.

2.2.5 DevOps and CI/CD

The term DevOps comes from the combination of development (Dev) and operations (Ops) processes. The Dev part focuses on software development and the Ops part on IT operations. The purpose of its existence is to solve potential conflicts between developers and operation specialists, who work in an agile software project, by following the DevOps approach [42]. In 2009, there was the first DevOpsDays event in Belgium, since when the DevOps plays a huge part in more agile software development, more time- and cost-efficient software production across companies [42]. This term is simply defined as a set of practices, tools, and a philosophy, which helps a software team, consisting of developers and operations specialists, to collaboratively work, trust each other and continuously improve themselves with the aim of optimizing and reducing the software development life cycle [42]. Continuous Integration and Continuous Delivery (CI/CD) practices are an important part of the DevOps process, which covers software development and IT operations to continuously build, test, release and deploy software in an automated manner [15]. The CI process is about merging source code changes into a main branch of a version control repository, then building and testing the software [42]. The CD process

enables the reliable and frequent software release after the building and testing steps, which ensures repeatable software deployment in case of decision for the deployment [22].

Relevance for this work: After adapting railway asset applications to the R4F Platform by using open-source interface standards, semantic web and OOP technologies, building, testing and delivering of these applications need to be automated and managed in a prototypical virtual environment to ensure the compatibility, reliability, consistency, traceability and comprehensibility of the applications in the platform while saving time, money and effort. For this, it is decided to continuously integrate and deliver the railway asset applications by utilizing the DevOps approach and CI/CD process, based on an automation workflow, in a virtual test environment for the research of this work (see [Subsection 1.7](#)). Therefore, the DevOps and CI/CD are seen as potential candidates to automate the AIP task for the Rail4Future project. Thus, these might answer the RQ1 in terms of AIP automation and management, and also the RQ2 regarding to striving for full-automation of the AIP (see [Subsection 1.4](#)). To demonstrate the AIP automation and management based on the DevOps and CI/CD, specific software technologies are applied to this environment, which significantly contributes to the proposed methodology (see [Subsection 4.3](#)). In this subsection, first, virtual machine (VM) is mentioned, which is necessary to lay out the test environment, representing a simplistic and compatible version of the R4F Platform, by providing software, and then apply the DevOps and CI/CD practices to the environment. Second, the software containerization technology is highlighted, which is important to implement CI/CD tools into the VM as isolated from each other. Then the pipeline technology is implied as a core element of the CI/CD process, where a CI/CD pipeline is used to realize the entire DevOps automation process in a defined workflow. After that, version control systems (VCS) are stated, that are used to trace and store the asset applications in a software repository with one or more branch collaboratively. Lastly, code analysis tools (CAT) are mentioned, which are also applied to the environment to detect vulnerabilities, bugs and redundancies in source codes of the railway assets.

2.2.5.1 Virtual Machines

VMs are a virtual version of a real machine having a different software so that these can handle cross-platform compatibility and hardware resource limitations. This helps to enhance the software portability and flexibility. [127]

From the system perspective, VMs have an instruction set architecture (ISA) providing an interface between the system and machine, which contain an application software (e.g., tools, libraries, OOP), OS (e.g., Linux, Windows) and hardware incl. input/output (I/O) resources, networking and a display [127].

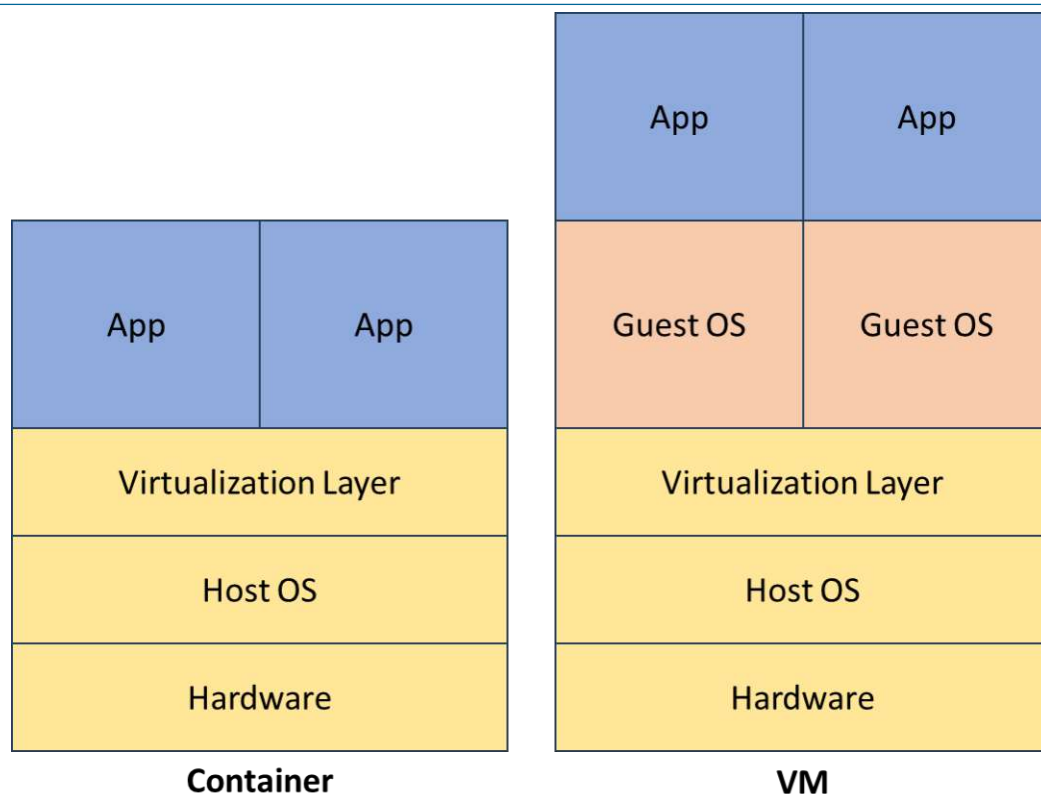
Relevance for this work: The above mentioned components are necessary to build the test environment in order to demonstrate the DevOps and CI/CD processes in the representative OS-compatible platform for the Rail4Future project as previously mentioned. Besides, VMs can keep files and folders, which are put through the file exchange between the real machine and the VM, through a file storage system. For the Rail4Future project, the files and folders, belonging to the asset applications of different railway subsystems, are directly stored and tracked in a VM. This VM is implemented as a framework of the environment, which is used to test the compatibility of these applications in the platform. Furthermore, a remote network tool supporting Secure Shell (SSH) connection between the real machine and VM is highly preferred for the Rail4Future project, because it provides a command line interface (CLI) window, helping the user to execute typical commands as well as shell scripts without utilizing any GUI, and a file explorer to manage and exchange the files and folders between the real machine and VM. Besides, the SSH connection is secure through the use of authentication tokens (e.g., username/password, public-private key pairs), and assists the user to login and control the VM remotely.

2.2.5.2 Software Containerization

Containerization in software engineering is a kind of virtualization technique, which provides a platform behaving like a VM with a separate OS and file system, and isolates processes and their resources from other platforms [115]. As Swartout et al. [131] mentioned, the containerization significantly reduces the overhead on contested hosts by only running a single kernel instead of multiple.

In Figure 2-2, a comparative overview of the structure of a software container and VM is shown. First, it is noteworthy that the VM runs a guest OS, which is the entire OS on top of the host OS. The guest OS is brought into the memory, which seizes several gigabytes storage and thus resources need to be partitioned. The VM is controlled by hypervisor, which helps to deploy one or more applications (app) using the hardware-based virtualization technology. Therefore, hypervisor is the virtualization layer of the VM and known as Virtual Machine Monitor partitioning and providing the VM with hardware (e.g., CPU, RAM). On the other hand, the container is operated only on the host OS as isolated from the guest OS making it more elegant compared to the VM. In this case, container runtimes are the virtualization layer of the container. Besides, the container can be put into the VM, it is also portable among several VMs. Moreover, a particular program and virtual environment can be identified in the container, which are immutable and able to be reproduced at any time. [146]

Figure 2-2: Structure comparison of a VM and container. [146]



Relevance for this work: Based on the above, software containerization is decided to be used to containerize CI/CD software tools, which are necessary

to enhance the DevOps-automated integration and processing of railway simulation assets in the test environment, for the Rail4Future project. By this, all the CI/CD tools are completely isolated from each other as software containers, which means reduced overhead on the VM host, belonging to the test environment. These tools are automation pipeline server, VCS and CAT, which are open-source and mentioned in next subsections in details.

2.2.5.3 Pipeline Technology

In the IT sector, the pipeline technology is a possibility to lay out a DevOps automation workflow, by which various software applications can be built, tested, delivered and/or deployed. To apply this key technology, there are certain popular software tools, which can also be containerized in a VM. Besides, the pipeline is able to be integrated with other technologies such as CAT and VCS helping to improve the CI/CD process. For example, Wadhams et al. [138] successfully automated the static code analysis process, helping to detect security vulnerabilities and performed by several CATs, using a CI/CD pipeline. Besides, Rayanagoudar et al. [109] used the VCS on their Jenkins CI/CD pipeline to track and store the version of their pipeline code script in a software repository of the VCS.

There are certainly challenges and limitations when applying the CI/CD pipeline technology to digital twins, which belong to different physical domains (e.g., automotive, railways, robotics). Zampetti et al. [143] defined and analyzed the challenges and limitations occurring during the use of the CI/CD pipeline in these domains. One of these domains is called railways, which is surely the main focus of this dissertation project among all the domains. They categorized these challenges and limitations in three: process-related challenges, barriers for CI/CD pipeline setting and maintaining, and pipeline-related challenges. They concluded that the configuration of the CI/CD pipeline highly depends on the domain. To find out the challenges and barriers, occurring in the railway sector, they interviewed an expert, who works as a software and hardware integrator in a large railway company and uses a CI/CD pipeline for CPS development. As process-related challenges the expert mentioned problems in onboarding developers regarding to the complexity of the railway domain, environment variability due to different trains, different interpretations coming from different developers, and thus expensive deployments. As pipeline-related barriers, the expert talked about limited availability of hardware and software

(e.g., preference of simulation environments due to a limited number of hardware test tracks). Finally, the expert named pipeline-related challenges such as static CAT configuration, long build execution time, continuous installation problems, lack of control over resources and network issues. Besides, the expert mentioned high costs, scalability issues, limited test automation and automated deployment on HiL related to the pipeline. Finally, the expert pointed out simulator specific challenges such as functional limitations, limited accessibility and failed interaction with the environment regarding to environment variability, which are related to the pipeline as well.

Relevance for this work: Use of the pipeline technology plays a crucial role in designing and implementing the CI/CD-based DevOps automation workflow of the AIP task for the Rail4Future project. In general, the technology ensures to build, test and finally deliver asset applications to stakeholders, which include different railway use cases (see [Section 5.1](#)) and are successfully implemented into the test environment, representing a simplistic and compatible version of the R4F Platform. Kugu et al. [77] successfully applied a CI/CD pipeline with the CAT and VCS to this environment and succeeded to automatically integrate and process the anti-slip traction and vehicle speed control co-simulation use case with this pipeline as well. This provides extremely valuable insights into the application of the DevOps practices with the CI/CD process for the completion of the Rail4Future project. [Subsection 4.3.2](#) clearly shows and describes the entire automation workflow, designed and implemented using a CI/CD pipeline for the Rail4Future project.

2.2.5.4 Version Control System

In recent couple of decades, the VCS technology becomes wide and popular to store, archive, exchange and track various kinds of files in software repositories among different internal and external stakeholders, who are owners of the repositories and thus authorized to deal with the files in their system. Surely, the user and role management plays a huge role in assigning the right roles to right users in the VCS, which is performed by officially authorized administrators of the VCS. By this, the data storage, update and exchange between these users are controlled to enhance a proper collaborative work environment, consisting of several VCSs and therefore several software repositories, between the stakeholders.

There are several VCS software tools, some of which are open-source, become extremely popular recently and are able to be easily integrated with other CI/CD tools incl. software tools providing CI/CD pipelines. As one concrete example, Rayanagoudar et al. [48] used the VCS technology to control the build of their software in a reliable and repeated way, which helps them to enhance their CI/CD pipeline with Jenkins BlueOcean, by committing their changes into the software located in a VCS repository. Besides Nandgaonkar and Khatavkar [88] found the VCS promising to track the version of their source codes, which are built, tested and then their contents are released using the CI/CD pipeline technology, for testing and development purposes.

Based on the above, the VCS technology is widely used for DTs, because these consist of data and models incl. simulation assets, which are to be tracked and stored regularly in VCS repositories belonging to different stakeholders. By this, they are able to control the digital version of a physical system in a virtual environment. For example, Jones et al. [67] studied challenges encountered while implementing the DT-based version control into product development. After their study, first, they pointed out difficulties in synchronization between physical and virtual artefacts of the product lifecycle management (PLM) during the design phase of the PLM. Then, they implied poor catching due to a great number of changes committed by stakeholders, which negatively impacts the fidelity of the version control in the PLM. Besides, the PLM processes (e.g., design, recycling) are hard to be followed because of changes in form and function of the virtual and physical artefacts according to their study. They mentioned remarkable limitations in comparison between two versions of the artefacts while quantifying and identifying changes, too (e.g., use of different materials and manufacturing methods). They also pointed out the necessity of the auto-capturing of the changes, which helps to decrease the manual effort to detect these changes.

Relevance for this work: For the Rail4Future project, the VCS technology is surely applied to the test environment, because this significantly helps to enhance and repeat the version control of source codes and files, belonging to the railway simulation assets, in a team collaboration as Kugu et al. [77] previously succeeded to manage the version control of these assets using the VCS in their CI/CD pipeline. The use of the VCS technology in the AIP automation and management is also mentioned in [Subsection 4.3](#).

2.2.5.5 Code Analysis

In recent decades, the OOP key technology, mentioned in [Subsection 2.2.4](#), is a huge trend to develop various software applications. For this, the necessity of the coding cannot be denied. Therefore, several source code scripts are created, updated and enhanced more by time, which may cause more complications in the code scripts. In general, code redundancies, bugs and security vulnerabilities typically occur after writing a bunch of classes, functions, attributes and variables in the scripts while using and applying a wide range of software packages, libraries and tools to the scripts. To overcome these complexities, the code analysis technology is highly preferred by many software engineers and researchers.

The technical information about the exact classification of the code analysis process can be found in [Appendix Subsection A.2.6](#).

In IT market, there are several software tools, designed and developed to perform a code analysis process in computer machines. Besides, it is a huge trend to integrate these tools with a CI/CD pipeline, which helps to apply the CI/CD process as an important part for DevOps practices as previously mentioned. By this, critical information (e.g., disclosure of credentials) about the source codes can be automatically extracted from the code analysis process within an automation workflow offered by the pipeline. For example, Zampetti et al. [142] investigated the impact of automated static CATs on CI pipelines. They pointed out the importance of being aware of potential errors, bugs and vulnerabilities during builds occurring in the CI process as an advantage of such tools. On the other hand, they suggested to check licenses (if licenses are wrong or missing), to specify tasks, to avoid unnecessary duplications and false positives, which potentially cause CI build slowdowns and breakages, using the CATs with the pipelines. Wadhams et al. [138] also implied sophisticated learning curve and the trade-offs between secure programming and workflow interruptions while implementing these static CATs within CI/CD pipelines for automated code analysis purposes.

Relevance for this work: For the Rail4Future project, the CAT significantly helps to clean up the source codes, used for the automation and management of the AIP task (see [Subsection 4.3](#)), while building and developing these. Kugu et al. [77] also succeeded to apply the code analysis technology to their CI/CD pipeline for the automation and management of the task in the test environment,

representing a simplistic and compatible version of the R4F Platform. As a result, they found it promising and useful to increase the code quality by automatically detecting potential issues in the source codes, and then handling these, which is essential for concise and secure code writing.

3 State of the Art

3.1 Literature Research Methodology

This subsection exposes the effective methodology, which is followed to do a comprehensive literature research in order to find out the state of the art of this work, namely in order to point out research gaps, which need to be filled by this thesis, and then extract scientific contributions and novelties of this work based on the comparison with other related research papers found out in the literature research.

A simplistic structured literature research approach for the state-of-the-art investigation of this dissertation is followed, which is based on a reasonable topic classification and selection criteria. These criteria, considered for picking a literature work, are as it follows:

- **Topic relevance**
- **Publication year**
- **Key technologies**
- **Corresponding sector(s)**

Table 3-1 shows an overview of the topic classification, considered for the structured literature research, with corresponding keywords used for the search process in popular search engines, some of which are Google Scholar, ScienceDirect, IEEE, MDPI and ResearchGate. First, simulations and digital twins for railways (SDTR) is mentioned as the first state-of-the-art subtopic of this dissertation work, which is related to a railway DT platform, called R4F, designed and developed for the Rail4Future project [108]. By this, the scientific relevance of this work to railway DTs with simulations is aimed to be extracted. Second, enabling simulations in digital twins (ESDT) is considered as a core state-of-the-art subtopic of this dissertation project, because this thesis is mainly about enabling simulations of particular railway use cases in a DT platform, which is the R4F Platform in this case, by providing a comprehensible, reliable and applicable methodology. This subtopic includes integration and processing of simulation assets (IPSA), related to the AIP task and methodology (see [Section 4.1](#) and [Subsection 4.2](#)), DevOps automation for simulations (DAS), which is important to automate and manage the AIP for more version control, quality control, easy-working, workflow-oriented control, time- and money-

efficiency at open-source level (see [Subsection 4.3](#)), and co-simulation implementation, helping to enhance use cases for more functionality (see [Subsection 4.4](#) and an example use case in [Subsection 5.1.3](#)).

Table 3-1: Topic classification for the state-of-the-art investigation.

Topic		Keywords
Simulations and Digital Twins for Railways		digital twin railway
Enabling Simulations in Digital Twins	Integration and Processing of Simulation Assets	simulation integration digital twin simulation model integration into digital twin single simulation fmi digital twin integrating simulations into digital twins ssp integrating simulation models into digital twins multibody simulation integration into digital twins
	DevOps Automation for Simulations	digital twin devops
	Co-Simulation Implementation	digital twin co-simulation

Two lists below indicate a total of 65 publications, found out for the state-of-the-art investigation, with key data such as topic correspondence, publication year, used technologies and sector(s) based on the selection criteria.

List 1: SDTR-related publications considered for the state-of-the-art literature research:

Nr. / Publication / Year / Key Technologies / Section incl. Subsystem

1. Digital Twin for Sustainability Evaluation of Railway Station (Kaewunruen & Xu [69]) / 2018 / DT, 6D BIM / Railways, railway station buildings

2. Digital twin aided sustainability-based lifecycle management for railway turnout systems (Kaewunruen & Lian [68]) / 2019 / DT, 6D BIM / Railways, railway turnout systems
3. Technology and Mathematical Basis of Digital Twin Creation in Railway Infrastructure (Shabelnikov & Olgezyer [123]) / 2020 / DT, Big Data, IoT / Railways
4. Adapting Digital Twin Technology in Electric Railway Power Systems (Ahmadi et al. [5]) / 2021 / DT, HiL / Railways, electric railway power systems
5. Digital Twins: An Advanced technology for Railways Maintenance Transformation (Dimitrova & Tomov [30]) / 2021 / DT / Railways, rolling stocks, tracks, catenaries, systems (e.g., passenger information system), bridges, ballast prism
6. The Fundamental Approach of the Digital Twin Application in Railway Turnouts with Innovative Monitoring of Weather Conditions (Kampczyk & Dybel [71]) / 2021 / DT, WS1 WiFi temperature measuring technology / Railways, railway turnouts
7. Fatigue damage assessment of complex railway turnout crossings via Peridynamics-based digital twin (Hamarat et al. [57]) / 2022 / DT, peridynamics / Railways, railway turnout crossings
8. Digital Twins for Managing Railway Bridge Maintenance, Resilience, and Climate Change Adaptation (Kaewunruen et al. [70]) / 2022 / DT, BIM / Railways, railway bridges
9. A Conceptual Model-based Digital Twin Platform for Holistic Large-scale Railway Infrastructure Systems (Zhou et al. [147]) / 2022 / DT / Railways, track, turnout, tunnel, bridge
10. Digital Twin for Railway: A Comprehensive Survey (Ghaboura et al. [44], Review Paper) / 2023 / DT / Railways
11. A monitoring based digital twin for the Filstal bridges (Lazoglu et al. [79]) / 2023 / DT, BIM / Railways, railway bridges
12. Concept for a digital twin of railway bridges on the example of the new Filstal bridges (Naranienci et al. [89]) / 2023 / DT, BIM / Railways, railway bridges
13. Performing Fatigue State Characterization in Railway Steel Bridges Using Digital Twin Models (Nhamage et al. [92]) / 2023 / DT, BIM, Fatigue Analysis System / Railways, railway steel bridges
14. Evaluating railway track stiffness using axle box accelerations: A digital twin approach (Shen et al. [126]) / 2023 / DT / Railways, vehicle-track interaction system

15. Revamping structural health monitoring of advanced rail transit systems: A paradigmatic shift from digital shadows to digital twins (Adeagbo et al. [3], Review Paper) / 2024 / DT / Railways, rail transit systems
16. Integration of Railway Bridge Structural Health Monitoring into the Internet of Things with a Digital Twin: A Case Study (Armijo & Sánchez [9]) / 2024 / DT, IoT, edge-cloud computing, ML / Railways, railway bridges
17. Augmented digital twin for railway systems (Bernal et al. [13]) / 2024 / DT / Railways
18. Digital twinning during load tests of railway bridges - case study: the high-speed railway network, Extremadura, Spain (Chacón et al. [20]) / 2024 / DT / Railways, railway bridges
19. Performance evaluation of electro-mechanical railway interlocking system for digital twin application (Chandaluri & Nelakuditi [21]) / 2024 / DT / Railways, electro-mechanical railway interlocking elements (e.g., crank handles, points, LC gate)
20. Digital Twins for Condition Assessment of Railway Infrastructures (Futai et al. [43]) / 2024 / DT / Railways
21. Improving safety management in railway stations through a simulation-based digital twin approach (Padovano et al. [100]) / 2024 / DT / Railways, railway stations
22. Digitalization of railway transportation through AI-powered services: digital twin trains (Sarp et al. [114]) / 2024 / DT, AI, IoT, Circular Economy, Big Data, Cloud Computing / Railways
23. Digital twin technology for continuously welded turnout on high-speed railway bridges based on improved MOPSO algorithm (Zhou et al. [145]) / 2024 / DT / Railways, high-speed railway bridges, railway turnouts

List 2: SDTR-related publications considered for the state-of-the-art literature research:

Nr. / Publication / Year / Topic Correspondence / Key Technologies / Section incl. Subsystem

24. From simulation to experimentable digital twins: Simulation-based development and operation of complex technical systems (Schluse & Rossmann [118]) / 2016 / IPSE // Industrial robotics
25. Experimentable digital twins for model-based systems engineering and simulation-based development (Schluse et al. [119]) / 2017 / IPSE // Industrial robotics
26. FMI and IP Protection of Models: A Survey of Use Cases and Support in the Standard / 2017 / CSI / FMI /

27. Digital twins: Understanding the added value of integrated models for through-life engineering services (Erkoyuncu et al. [36]) / 2018 / IPSE / ROS, CAD, EDA Software / Mobile robots
28. FMU-supported simulation for CPS Digital Twin (Negri et al. [91]) / 2019 / IPSE / FMI / Production systems
29. Multi-modelling and Co-simulation in the Engineering of Cyber-Physical Systems: Towards the Digital Twin (Fitzgerald et al. [37]) / 2019 / CSI / FMI, SysML / CPS, robotics
30. Digital twin and virtual reality: a co-simulation environment for design and assessment of industrial workstations (Havard et al. [61]) / 2019 / CSI / FMI, CPPS, robotics
31. Real-time co-simulation for the virtual commissioning of production systems (Scheifele et al. [116]) / 2019 / CSI / FMI, API / Production systems
32. Co-simulation as a Fundamental Technology for Twin Ships (Hatledal et al. [59]) / 2020 / CSI / FMI, SSP, OSP / Maritime industry
33. Digital Twin Modelling of Ship Power and Propulsion Systems: Application of the Open Simulation Platform (OSP) (Perabo et al. [103]) / 2020 / IPSE, CSI / FMI, OSP / Maritime industry
34. TwinOps-DevOps meets Model-Based Engineering and Digital Twins for the engineering of CPS (Hugues et al. [64]) / 2020 / DAS / CI/CD, Pipeline, Software Containerization / Cyber Physical Systems (CPS)
35. Towards a DevOps approach in Cyber Physical Production Systems using Digital Twins (Ugarte et al. [136]) / 2020 / DAS // Cyber Physical Production Systems (CPPS)
36. The Potential of FMI for the Development of Digital Twins for Large Modular Multi-Domain Systems (Wiens et al. [140]) / 2021 / IPSE / FMI / Renewable energy, wind energy
37. The Digital Twin as a Common Knowledge Base in DevOps to Support Continuous System Evolution (Mertens & Denil [84]) / 2021 / DAS // CPS, robotics
38. Accurate Robot Simulation for Industrial Manufacturing Processes using FMI and DCP Standards (Shah et al. [124]) / 2021 / CSI / FMI, DCP / Robotics
39. Virtual Hardware-in-the-Loop FMU Co-Simulation Based Digital Twins for Heating, Ventilation, and Air-Conditioning (HVAC) Systems (Abrazeh et al. [2]) / 2022 / IPSE, CSI / FMI, Deep Reinforcement Learning / Heating, Ventilation and Air-Conditioning Systems (HVAC)
40. Towards Digital Twin-enabled DevOps for CPS providing Architecture-Based Service Adaptation & Verification at Runtime (Dobaj et al. [32]) / 2022 / DAS // Industrial Product-Service Systems (IPSS), CPS

41. FMI real-time co-simulation-based machine deep learning control of HVAC systems in smart buildings: Digital-twins technology (Mohseni et al. [87]) / 2023 / IPSE / FMI, Deep Reinforcement Learning / HVAC systems
42. Hybrid Digital Twins Using FMUs to Increase the Validity and Domain of Virtual Commissioning Simulations (Pfeifer et al. [104]) / 2023 / IPSE / FMI / Production systems
43. Model-Based DevOps: Foundations and Challenges (Combemale et al. [23]) / 2023 / DAS // CPS
44. A Machine-Learning-based Surrogate Modeling Methodology for Submodel Integration in the Holistic Railway Digital Twin Platform (Zhou et al. [149]) / 2023 / IPSE / FMI / Railway infrastructure system
45. Leveraging semantic technologies for the application in multi-domain digital twins (Kern [72]) / 2023 / IPSE / FMI, ontology, OWL / Railway infrastructure system
46. Towards DevOps for Cyber-Physical Systems (CPSs): Resilient Self-Adaptive Software for Sustainable Human-Centric Smart CPS Facilitated by Digital Twins (Dobaj et al. [33]) / 2023 / DAS / Industrial IoT, Command Line Tool (CLT), SSH / CPS
47. Digital twin and agent-based simulation: co-simulation to support intelligent navigation of healthcare mobile robot (Anyene et al. [7]) / 2023 / CSI / Unity3D, ROS / Healthcare, robotics
48. Fault Injection in Co-simulation and Digital Twins for Cyber-Physical Robotic Systems (Larsen et al. [78]) / 2023 / CSI / FMI / CPS, robotics
49. Co-simulated digital twin on the network edge: A vehicle platoon (Palmieri et al. [101]) / 2023 / CSI / FMI, 5G / CPS, vehicle, network
50. Co-simulation Digital Twin Framework for Testing Future Advanced Air Mobility Concepts: A Study with BlueSky and AirSim (Zhao et al. [144]) / 2023 / CSI / MATLAB/Simulink, TCP/IP / Aircraft systems, Unmanned Aircraft System (UAS)
51. Integrating FMI and ML/AI models on the open-source digital twin framework OpenTwins (Infante et al. [65]) / 2024 / IPSE / FMI, ML /
52. Towards a Standardized Framework for Collaborative Ship Powertrain Design using the System Structure and Parameterization Standard (Neelamraju [90]) / 2024 / IPSE, CSI / FMI, SSP / Maritime industry
53. A Reinforcement-Learning-based Parameter Tuning Methodology for Traction Control in the Holistic Railway Digital Twin System (Zhou et al. [148]) / 2024 / IPSE, CSI / FMI, SSP / Railway infrastructure system
54. A DevOps Approach for the Systematic Development and Evolution of Built Assets Digital Twins (Aissat et al. [6]) / 2024 / DAS / BIM, VCS, CI/CD pipeline, software containerization, microservices / Building, GRIDD Lab Room

55. Digital Twin Prototypes for Supporting Automated Integration Testing of Smart Farming Applications (Barbie et al. [11]) / 2024 / DAS / CI/CD pipeline, software containerization / Agriculture, smart farming applications
56. Automating build, deployment, and monitoring of model-based digital twins (Markusfeld [82]) / 2024 / DAS / Cloud infrastructure, CI/CD pipeline /
57. Design and implementation of an integrated DevOps framework for Digital Twins as a Service software platform (Scherma [117]) / 2024 / DAS / CI/CD pipeline, VCS, API, UI /
58. Digital-twin co-simulation framework to support informed decision in healthcare planning and management (Anyene et al. [8]) / 2024 / CSI / Unity3D, ROS / Healthcare, robotics
59. Development of a Digital Twin for prediction of rail surface damage in heavy haul railway operations (Ahmad et al. [4]) / 2024 / CSI / / Railway infrastructure system, rail vehicle and track
60. A Study on Co-simulation Digital Twin with MATLAB and AirSim for Future Advanced Air Mobility (Turco et al. [135]) / 2024 / CSI / / Aircraft systems, UAV operations
61. A Co-simulation Digital Twin with SUMO and AirSim for Testing Lane-based UTM System Concept (Wen et al. [139]) / 2024 / CSI / / Aircraft systems, UAV operations
62. Model-Based Control for Power-to-X Platforms: Knowledge Integration for Digital Twins (Dittler et al. [31]) / 2025 / IPSE / FMI, SSP, semantics, knowledge graph / Energy systems
63. Agile Continuous Integration Testing of Embedded Software Systems with Digital Twin Prototypes (Barbie [10]) / 2025 / DAS / CI/CD pipeline, VCS, CAT, VM, software containerization / Embedded software systems, smart farming applications
64. Streamlining digital twin development and operation with DTOps (Miao et al. [85]) / 2025 / DAS / CI/CD pipeline, VCS, graph database / Gear production line
65. Continuous integration and continuous deployment for cyber-physical systems by graph based meta data models (Reiterer [110]) / 2025 / DAS / CI/CD pipeline, graph database / CPS

3.2 Topic Classification

3.2.1 Digital Twin for Railway Infrastructure Systems

Since many years, the use of the DT technology is widespread among railway stakeholders and researchers, who work for monitoring, operation,

maintenance, research and analysis of a railway infrastructure system. The research findings, listed in the List 1 above, are sorted according to subsystems of the infrastructure system as it follows:

- **Railway Turnouts:** Kaewunruen & Lian [68] created a 6D BIM of railway turnouts, which helps to predictively analyze, maintain and visualize these turnouts collaboratively by using 3D and 4D (3D model with schedule) simulations in the DT-based 6D BIM as they mentioned. In addition, Kampczyk & Dybel [71] worked on a DT application, using the WS1 WiFi temperature measuring technology, to monitor weather conditions of railway turnouts, significantly affecting the safety in railway infrastructures. Besides, Hamarat et al. [57] performed Peridynamics simulations in their DT for fatigue damage analysis of railway turnout crossings. Furthermore, Zhou et al. [145] proposed a DT, where they used a Virtual Entity (VE) model, based on FEM simulations, to evaluate track structures and continuously welded turnouts, implemented on high-speed railway bridges, in a virtual world. Even, they mentioned that the DT technology leads to collaboratively, virtually inspect and operate the entire physical system overcoming the restrictions of the system such as limited measurement data due to predictions, and the lack of a reliable and interactive connection between the data and current structural situation. In addition, they suggested to apply ML-based surrogate models to these FEM simulations, which take relatively much time, to accelerate the simulation process in the DT.
- **Railway Bridges:** Kaewunruen et al. [70] developed a DT platform using the BIM technology for life cycle and visual analysis of a railway bridge, that surely helps to extensively operate, maintain the bridge, and manage its assets reliably and sustainably in spite of spontaneously occurring constraints such as limited component persistence (e.g., corruptions) and lack of real-time conditional monitoring. Besides, Lazoglu et al. [79] and Naraniecki et al. [89] proposed a DT-based BIM for Filstal railway bridges in Germany, which includes measurement data, visualization of the bridges, and is utilized for their inspection and structural safety verification. In addition, Lazoglu et al. [79] defended the idea of using computational simulations in the DT concept, helping to evaluate the condition and structural safety of the bridges as compared with measurements. Furthermore, Nhamage [92] proposed a BIM-based DT, which is used for calculation of fatigue damage of railway steel bridges. They also used numerical simulations for the fatigue damage calculation in their DT to imagine different fatigue train scenarios, occurring in these bridges (e.g., standard vs. heavy traffic load), in the virtual environment. Besides, Chacón et al. [20] demonstrated realistic

load scenarios, occurring in railway bridges of a high speed train network in Extremadura, Spain, and deployed during their load tests helping to determine their structural concerns and load capacities in terms of their structural integrity and safety, by utilizing numerical simulations in their digital twinning approach. As they pointed out, the DT technology leads to make digital assets incl. the simulations available and sharable between different stakeholders by providing an IT and communication framework, which also allows to analyze static and dynamical behavior of physical assets such as the railway bridges for their maintenance needs. What is more, Armijo & Sánchez [9] suggested a DT approach in conjunction with IoT, ML and on-premises-cloud hybrid computing technologies allowing structural health monitoring (SHM) of railway bridges by using ML-based algorithms. These algorithms are the enhanced version of typical physics-based simulations used to accurately detect potential damages in the bridges while reaching better simulation performance. In addition, they mentioned advanced physics simulations, which they would utilize in their DT for more comprehensive damage detection of the bridges increasing the fidelity of the DT.

- **Railway Vehicle and Track:** Dimitrova & Tomov [30] named advantages of implementing simulations into a DT such as predictive fault analysis and detection of vehicles and tracks for better railway maintenance. Besides, Zhou et al. [147] adapted an application, containing Vehicle and Track Interaction (VTI) simulations used to predict track geometry for smart maintenance of railway tracks, to their conceptual R4F DT Platform for the Rail4Future project [108]. Moreover, Shen et al. [126] suggested a DT framework for evaluation of railway track stiffness, purposing track conditions monitoring and maintenance, by using physics-based simulations of a Finite Element (FE) VTI model. Furthermore, Adeagbo et al. [3] did a comprehensive literature survey to explore the potential of the DT technology for more optimal SHM of advanced rail transit systems. They named simulation techniques such as discrete-event simulation, FEM, CFD, etc. according to their exploration. They also found out that the simulations in DTs can be used for predictive analysis, visualization, model training (e.g., surrogate models, for an example see Zhou et al. [149]) and inferencing purposes. As they concluded, the DT is a highly promising technology to supply more consistent and up-to-date information about the health and performance of the rail systems. While talking about model training, Bernal et al. [13] proposed an augmented DT, which is defined as an AI-powered DT with the highest maturity level [243], and implements a surrogate model into an MBS numerical model of a railway vehicle,

interacted with a track, to predict the vehicle derailment risk for empty and loaded cases.

- **Other Subsystems:** As an example, Kaewunruen & Xu [69] successfully converted the 3D model of a railway station into a 6D BIM, which is based on DT and utilizes information modelling simulations used for renovation management, time-scheduling, estimation of life cycle costs and carbon emissions. Besides, Padovano et al. [100] performed pedestrian simulations in their prototypical DT to study the dynamical crowd effect, caused by pedestrians, on their comfort and safety in railway stations. By this, the DT platform helps railway station managers to manage the crowd by predicting the crowd flow, planning evacuation reaction or gate openings/closings in the stations in a well-informing and in-time decision-making manner as they pointed out. As another example, Ahmadi et al. [5] proposed an approach to implement a DT into an electric railway power system, and pointed out exclusive benefits of the DT application for the system such as conditional monitoring, timetable management, energy management, power flow analysis, power quality monitoring, operating profile optimization, fault diagnostic and maintenance. Additionally, they suggested to use power system and real-time HiL simulations in their railway DT as an auxiliary element to analyze the system behavior, and to find out uncommon situations incl. faults in the entire system, which is important for a safe and trustworthy operation of the physical system. Shabelnikov & Olgezyer [123] also pointed out simulation of uncommon events, coming from a railway humping use case, as a possibility of the DT use for the railway sector. Moreover, Chandaluri & Nelakuditi [21] created a DT of electro-mechanical railway interlocking systems (e.g., crank handles, points, LC gate), where they simulated these systems and verified their functionality for high-level safety of train movements. They also talked about time- and cost-efficiency as a result of automated data flow between the virtual and physical systems, offered by the DT technology.

Surely, the existence of potential challenges and limitations during the design and development of a DT platform can't be ignored. Ghaboura et al. [44] successfully identified and argued these challenges and limitations behind the scenes regarding to safety, security, data, representation and intelligence during their comprehensive review study, even if they noticed the high potential of the DT technology for an optimal digitalization and automation of the railway domain. First, they implied the need for real-time, secure data access and stream from sensors. Then, they referred to the integration and synchronization issues of different railway systems, and thus the lack of a standardized and

extensive methodology (e.g., data integration problems). Moreover, they pointed out challenges in design and development of AI models, which takes much time, costs and works with limited data on the one hand and on the other hand, these AI models need to be efficient and reliable. Bernal et al. [13] also indicated the necessity for a careful design of an ML-based surrogate railway vehicle model, and its strict dependency on training datasets in order to properly implement the model into their augmented DT for railway applications, which they developed for their work. Furthermore, they indicated the need of high redundancy and effort for creation of virtual models belonging to the railway infrastructure system (e.g., 3D model of a controlling room). Besides, virtual inspection through the use of the models in a consistent and stable platform, shared among railway stakeholders, is yet to be achieved as they found out. In addition to that, they pointed out the research gap between the virtual and real spaces, where enabling feedback loops through the use of VR and XR technologies is an essential topic and thus worthy of investigation as they mentioned. Furthermore, they drew the attention to the essence of full realization and evaluation of the data privacy and security using secure, reliable and persistent methods and technologies in building DTs for railways. Sarp et al. [114] also pointed out cybersecurity and privacy concerns in their comprehensive review about the railway digitalization, based on AI-powered services, due to increasing demand on data exchange and communication in DT train systems. They suggested to reinforce data encryptions, access control and authentication facilities, and follow secure protocols and laws concerning passengers' privacy (e.g., GDPR). Furthermore, Adeagbo et al. [3] emphasized the need for more collaboration between different DTs resp. stakeholders, which helps to gain better scalability and complexity tolerance, the essence for overcoming high complexity, caused by these several DTs incl. multiple systems and key players, the need for unifying the DT architecture solution, consisting of a single design, platforms and tools, and the dealing with the concern regarding to discrete information systems, caused by data fragmentation. In addition, they mentioned ownership, ethical and copyright issues occurring during data exchange in the DTs. What is more, Futai et al. [43] pointed out technology-dependent challenges and innovations such as the entire infrastructure management by the DTs incl. monitoring, inspection and maintenance tasks, cooperative cloud computing in the DTs for data acquisition management purposes, conversion of inspection into monitoring (e.g., SHM substitution), computer vision implementation for damage detection and

monitoring, auto alarming systems for warning about ML-based data feeding, and structural DT conception using sensors and inspection data for diagnosis and monitoring of the physical infrastructure system. Lastly, Ghaboura et al. [44] named limitations, encountered in their comprehensive review, such as the lack of literature research on Haptics, limited development of a standardized and comprehensive framework and its proper integration with key technologies such as Metaverse, Haptics, Reusable AI and Blockchain, and the need for an extended recognition of more DT-related publications in the review, even if they don't mention the railway domain.

3.2.2 Enabling Simulations in Digital Twins

3.2.2.1 Integration and Processing of Simulation Assets

Since a few decades, using simulation assets in DTs gains a huge trend to predictively analyze, maintain, validate and visualize physical systems in research and industry. However, research gaps between these simulation assets and DTs come into being while trying to integrate and process these in the DTs, because these assets might possess high data complexity, bugs, security breaches, multiple versions, require high computation time and effort, be defect, non-working, tool dependent, platform incompatible, lack of descriptions, validations, optimizations, parameters, connections or connectors. This dissertation work first aims to fill the research gaps between simulation assets of railway use cases (see [Subsection 5.1](#)), and the R4F DT Platform by providing the AIP methodology (see [Subsection 4.2](#)) as previously mentioned. Therefore, IPISA-relevant research findings, purposing to close research gaps between DTs and simulations, are found, listed in List 2, and mentioned in this subsection.

Schluse & Rossmann [118] have seen the DT as a potential solution to put different aspects across different dimensions (e.g., users, methods, simulation domains...) together as interacted with each other when utilizing simulations of a physical system. This also helps to decrease the effort for creation of different simulation applications, to provide several use of these, rapid switches between these, and to further build some of these, which cannot be imagined so far as they mentioned. Besides, they pointed out the necessity of designing and developing methods and approaches to integrate the dimensions incl. the simulation applications in the DT. Besides, Schluse et al. [119] successfully applied the DT technology to Model-Based Systems Engineering (MBSE),

where they aimed to fill the gap between systems engineering, which is an interdisciplinary area dealing with the design, integration and management of complex engineering systems (e.g., robotics) over their life cycle, and simulations by using a DT. Moreover, Erkoyuncu et al. [36] proposed a common model space, where they built a DT of a mobile robot by using Robot Operating System (ROS), to integrate a robotic simulation model in the common space. In conclusion, they pointed out the importance of the model integration for enhancement, real data-based growth of simulation-based design, and higher simulation fidelity by using the DT. They also mentioned the need for connection between the model and its common space helping to ease the model integration task incl. user input parameterization. Furthermore, Negri et al. [91] suggested to use the open-source FMI standard for CPS DTs to easily, environment-independently exchange and reuse various black-box modules of a core simulation model, which contain specific behaviors, in different virtual environments such as the DTs. Perabo et al. [103] also supported the idea of the FMI usage for integration and simulation of maritime subsystems (e.g., AC ship power and propulsion system) as FMUs, which can later be parameterized based on scenarios, configured for simulation (e.g., change of simulation step size) and connected to each other for co-simulation purposes, in a maritime DT. What is more, Wiens et al. [140] suggested to apply ontologies to effectively manage the knowledge about interfaces, provided by the FMI standard, between simulation models and DTs. In addition, they pointed out the need for further standardization in the development of the models, helping to faster connect these to each other in terms of co-simulation, as well as the necessity of model reduction strategies to add real-time abilities to DTs (e.g., ML-based surrogate modelling technique proposed by Zhou et al. [149]). Speaking about model reduction strategies, Mohseni et al. [87] proposed a real-time ML-based control methodology, and integrated this into their HVAC DT by using the FMI for more economical energy management of a HVAC system. Besides, Zhou et al. [149] and Zhou et al. [148] also proposed to apply the FMI and SSP standard in conjunction with other software technologies (e.g., VCS repository, graph database) to integrate and process their AI-based ML applications in the R4F DT Platform. Furthermore, Abrazeh et al. [2] proposed to use the FMI to tool-independently integrate HiL (physical controller) and SiL (virtual controller) as coupled in a DT of an HVAC system, aiming to control its temperature and humidity by providing an adaptive control methodology. Additionally, Pfeifer et al. [104] successfully built a hybrid DT, consisting of a virtual commissioning

component, used to design and validate a control system of a production plant, and an elastic multibody systems model. As they pointed out, this hybridization helps to combine the benefits of both these models (machine + control), and thus demonstrate the validation of the elastic multibody systems model in a virtual environment after integrating these models into their DT (incl. easy and time-saving parameterization of simulation) through the use of the FMI. What is more, Infante et al. [65] developed an open-source framework to integrate different simulation features into DTs, to manage and validate default and FMI-based simulation services in the DTs by using ML models and simulation standards such as the FMI after they noticed the lack of such a framework as a considerable limit for the integration of these features into DTs. As basics of the FMI validation, Bertsch et al. [14] also discussed the necessity to evaluate the FMI standard from an industrial perspective by using the FMI Compliance Checker and FMI Cross Checking rules, which help to gain insights into the quality of an FMU file and FMI implementation, and thus to increase the maturity level of FMI-based simulations. In addition, Infante et al. [65] used a database to effectively store and manage the entire data, utilized and generated by a simulation process itself. Beside the use of the FMI, Neelamraju [90] pointed out the high potential of the SSP standard for a standardized system description, model interoperation and exchange in conjunction with an ontology, as demonstrated in an example of a ship powertrain DT framework belonging to the maritime sector. Speaking about the ontology term, Dittler et al. [31] created and developed an ontology-based approach, aiming to adaptively control any process (e.g., simulation) by effectively integrating the knowledge about simulation models, their FMUs, interrelations and other simulation-relevant data into a DT. Kern [72] also created and developed the R4F Ontology to increase the traceability and comprehensibility of different railway simulation assets by demonstrating a knowledge integration through the use of the semantic technology with the FMI standard in the R4F Platform (see [Appendix Subsection A.2.5](#)).

3.2.2.2 DevOps Automation for Simulations

Since several years after proposing DTs, a trend on their use for DevOps of physical systems is rapidly growing, which provides virtualization and automation solutions. Nowadays, simulation applications of the physical systems are tended to be continuously and automatically built, tested, released and deployed by using the DevOps approach to increase their quality, reliability

and fidelity in the DTs, which might include the auto-integration, -processing of the simulations, and simplification of its management in a DT of a physical domain. This dissertation also covers the automation and management of the AIP task in the R4F Platform by applying DevOps practices to simulation asset applications in the platform as previously mentioned. Therefore, DAS-relevant research example works, which purposes to close the gap between DevOps automation, simulations and DTs, for the state-of-the-art investigation of this work's research are found, listed in List 2, and revealed in this subsection.

Ugarte et al. [136] saw the DT as a high potential to apply DevOps practices in CPPSs, which help to automate the development and deployment of CPPSs, and fill the gap between development, incl. the use of simulations to partially verify engineering systems such as CPPSs, and operations. Besides, Mertens & Denil [84] preferred to apply the DT technology as a common knowledge base to the DevOps loop of a CPS, which is closed by the DT helping to gain more consistent simulation results by providing continuous feedback from monitoring of operations of the system to development of the system as they found out in a simple robotics use case example. Moreover, Hugues et al. [64] successfully combined the DevOps approach with the DT and Model-based Engineering terms in the name of TwinOps, which they proposed as a process providing a modelling, simulation, validation and monitoring framework based on a workflow. For the demonstration of this process in a CPS DT, they suggested to use a CI/CD pipeline, helping to automate every step in the workflow, and software containers to construct a replicable modelling environment while promoting interface standards such as the FMI for co-simulations and exchange of simulation models in the DT environment. Furthermore, Dobaj et al. [32] mentioned the necessity of combining both the development (Dev) DT, containing the simulation workspace, and operations (Ops) DT, representing the physical reality, to decrease the risks, caused by model and simulation instabilities, during CPS operations for verification purposes. Beside that, they proposed to use CI/CD pipelines to increase the automation level of DevOps processes (e.g., software build, test, deploy) in CPS. By the way, Dobaj et al. [33] applied the CI/CD pipeline technology to their DT-based self-adaptive software framework, which they proposed to evaluate their software system implemented in a CPS, to build an automation workflow consisting of real-time data processing. As an example, they integrated a timing model of a cyclic real-time processing pipeline into their DT to monitor and control the timing behavior

of a DT-enhanced control service, used for operation and management of CPS. By the way, they preferred to use a machine, representing the external management system, for accessing the DT through SSH connections to interactively, experimentally, remotely control and monitor the DT-based control service. What is more, Combemale et al. [23] mentioned the Model-based DevOps approach, which aims to investigate the auto-connections of CPS design models to CPS runtime monitoring as DTs. Thus, it helps to simulate reconfigurations for optimization of production processes, to gain insights from simulations about resource consumption and emissions, and to transparently provide information about operations. They also pointed out its challenges, which concern the controlled software updates, management of the connections by applying DTs into DevOps, and automated extraction of Dev-to-Ops-to-Dev models for continuity between Dev and Ops. Besides, Aissat et al. [6] proposed a DevOps approach to build, test, deploy and operate DTs, which might be a software system and include simulations representing various scenarios in a digital environment. In the meanwhile, they suggested to use VCS, which can seamlessly interact with CI/CD pipelines, to store and track project artifacts incl. source codes, configurations and documents in a team collaboration, which significantly supports the entire DevOps process. Beside that, they applied a CAT to the CI/CD pipeline for static code analysis purposes, which might include detection of security vulnerabilities (e.g., user credentials), bugs or redundancies in source codes. Speaking about simulations, they mentioned interoperability concerns regarding to building dynamic models, which represent built assets, while developing DTs for these assets. Furthermore, Barbie et al. [11] demonstrated the automated integration testing and development of an embedded software system without connecting the system to any physical asset by applying DevOps techniques, incl. continuous integration and continuous delivery processes using CI/CD pipeline, VCS, software containerization technologies, to a DT of a smart farming application. Speaking about software containerization, they marked significant benefits of containerizing the embedded software systems such as platform-compatible software development and deployment, machine setup independency and process isolation while providing communication with other layers using specific tools. In the meanwhile, they mentioned the need for simulations to supply a virtual representation for the DT, although it causes communication-based complexities for the development of the software system in the virtual environment. In conclusion, they pointed out the independency from any costly

physical hardware, the possibility of software integration into CI/CD pipelines using open-source tools, and the demonstration of various scenarios and conditions, the real-life imitation of which might be challenging, as advantages of the DT use. Besides, Barbie [10] designed and developed a DT concept for Industry 4.0 applications, which is used for development, testing, simulation, operation and monitoring of CPS in conjunction with DevOps practices incl. utilization of CI/CD pipelines, static CATs, containers, VCS, databases for the automation of the tasks in the name of Industrial DevOps. In addition, Markusfeld [82] utilized the DevOps pipeline technology to automatically create DSL-based models, which are system-dependent models created in a DT platform specific language and thus called model-based DTs as an example, from GPL-based models, built in or between several systems, then deploy and operate these DTs. Moreover, Scherma [117] proposed a DevOps framework to apply it to a DT, which is presented as a Service web application with implemented UIs, for its lifecycle management by providing VCS and CI/CD pipeline technologies into this framework. The DT platform might include VMs, containers, physical servers or cloud-based machines as she shortly mentioned. What is more, Miao et al. [85] successfully adapted DevOps methods into a service-oriented DT system, which they proposed and called DTOps (DT + Operations). By this, they promised continuous, feasible and efficient integration, development, delivery and operation of the DT system, which may include digital assets such as simulation and prediction of system behavior providing insights into technical problems (e.g., system defects) of a physical system. Beside that, they used graph databases to extract information about all assets (incl. the digital assets) and their interrelations from dependency graphs. Speaking about graph databases, Reiterer [110] proposed a graph-based metadata modelling approach, which is used to develop DevOps processes by auto-generating build and deployment pipelines, and descriptions of simulations. By this, he aimed to increase the traceability, reusability and comprehensibility of the simulation applications. In addition, he applied the graph-based methodology to the MBSE, DCP and SSP standards. He also suggested to apply it to DTs in future.

3.2.2.3 Co-Simulation Implementation

Beside integration and processing of available simulation models and data into DTs, enhancement of these models by cooperatively simulating these with other subsystems or models, namely co-simulation of these models may be an

interesting and trending way to gain more functionalities from these available models, which represent a physical system (e.g., vehicle), helping to imagine further realistic scenarios in a virtual environment. As previously mentioned, this dissertation work covers the implementation of co-simulations into the R4F Platform (for a use case example see [Subsection 5.1.3](#)). Therefore, CSI-related research findings are explored, listed in List 2, and stated in context in this subsection.

Fitzgerald et al. [37] suggested to use the FMI standard to control a robot by co-simulating its discrete motors, which are packaged into FMUs and then implemented into a model-based DT framework as connected to each other, as a demonstration CPS example. Besides, Havard et al. [61] succeeded a real-time co-simulation workflow between a DT and VR environment to imagine realistic system behavior of a CPPS in a 3D environment by providing an architectural solution while utilizing the FMI to ensure and manage the data flow between the simulations in the DT, and the VR in a co-simulation architecture. Beside that, Scheifele et al. [116] successfully integrated real-time co-simulations, used for virtual commissioning of a production system, into a DT of the system by using model integration interfaces such as the FMI. Moreover, Hatledal et al. [59] proposed an open-source framework to build a DT of a maritime system, aiming to analyze, maintain it, and predict its system behavior by using multiple simulation models as co-simulated. They saw the co-simulation as a solution to overcome concerns regarding to integration of different systems, hardware and software resources making difficult to implement the DT in a centralized and homogeneous way. As they noticed, the co-simulations are hard to be adapted to a DT because of the system heterogeneity, caused by the system multiplication, which can be solved by utilizing related standards and tools such as FMI, SSP, DCP and OSP software, mentioned in [Subsection 2.2](#), in their framework. Besides, Zhao et al. [144] found out potential benefits of a co-simulation implementation into a DT such as enhanced, accurate and combined simulation facilities after they proposed a co-simulation DT framework and demonstrated a co-simulation example in the framework for automated UAS operations in the aircraft sector. On the other hand, they mentioned the necessity for a standardized and universal co-simulation framework and communication interface to provide a compatible, synchronizable and interoperable virtual environment. This might be achieved using open-source interface standards and software tools, which are mentioned

above. After that, Wen et al. [139] and Turco et al. [135] suggested further co-simulation methods to provide more standardization, interoperability, compatibility and thus functionality in the DT-based co-simulation framework. What is more, Perabo et al. [103] suggested to model a DT of an electric ship power and propulsion system by utilizing the OSP software application, by which they co-simulated FMUs of ship power and control elements, belonging to the system, in a DT- and OSP-based co-simulation environment. Furthermore, Abrazeh et al. [2] successfully integrated the HiL (physical controller) and SiL (virtual controller) models into a co-simulation framework by exchanging and co-simulating these as FMUs to design and develop a DT-based control model used to adaptively manipulate the temperature and humidity of a physical HVAC system. By this, they showed the crucial benefit of the co-simulation implementation for enhancement and improvement of a functionality in a physical domain by interoperating different subsystems. Anyene et al. [7] also proved it by combining the DT concept and simulations in a co-simulation framework, which they provided as a solution for intelligent navigation of mobile robots in hospital environments as a concrete demonstration example. Beside that, Anyene et al. [8] successfully integrated a DT of a hospital physical twin with discrete event co-simulations within a co-simulation framework, which supplies real-world scenarios (e.g., patient-care scenarios) and is used for informed decision-making purposes in healthcare planning and management. What is more, Larsen et al. [78] used a Co-Simulation Orchestration Engine to co-simulate multi-formalism FMU models, which contain components of a CPS, in a DT of the system for fault injection in the components allowing to analyze various kinds of faults and failures in a virtual environment. Beside that, Palmieri et al. [101] proposed a co-simulation approach for a multi-domain CPS DT, consisting of a physical, control and communication models. They also mentioned significant advantages of the co-simulation implementation such as system complexity management, flexible and modular simulation environment, and component reusability while the FMI provides isolation and interoperability of different components. In addition, they promoted the distributed co-simulation implementation as an alternative solution to enable a scalable co-simulation framework by distributing the entire computation effort, made for simulation of large and complex systems, across multiple machines for their future work. As a concrete example of the distributed co-simulation, Shah et al. [124] demonstrated a robotic arm co-simulation for industrial manufacturing robots in a distributed fashion within a virtual

environment by utilizing the FMI and DCP standards. By the way, they promoted the DCP standard providing a master-slave communication architecture for the distributed co-simulation implementation, where they embedded DCP master and slaves, representing different simulation models, tools and systems, into FMUs for more compatibility, accuracy and interoperability between different machines, tools, models and software. In a distributed co-simulation implementation, it is important to consider the aspects about the IP protection of simulation models during their distribution into other stakeholder's machines or workstations. Durling et al. [34] pointed out these concerns based on some use case examples, which they collected for their study, while sharing FMUs between different stakeholders. Besides, Hatledal et al. [60] proposed an FMI-based co-simulation framework, which is open-source, language-, platform-independent, and might be implemented into a DT platform, to solve the IP leaking issue. What is more, Ahmad et al. [4] succeeded to co-simulate an MBS railway vehicle model with a traction model in a railway DT framework to realize train behaviors, based on wheel-rail friction and traction, for detection of rail surface damages. In conclusion, they pointed out the need for high computation effort, caused by the MBS model, and a reliable communication between different software tools and interfaces, used for co-simulation purposes. As they suggested, the high computation issue would be solved by decreasing the complexity of the MBS model, an example of which Zhou et al. [149] also demonstrated by proposing an ML-based surrogate modelling approach. Beside that, they determined better prediction results coming from the DT technology compared to the Digital Shadow approach. Speaking about predictions, Zhou et al. [148] applied an RL-based parameter tuning methodology to a co-simulation model, which aims to couple an MBS vehicle and PID controller model for anti-slip traction behavior and vehicle speed control, and is successfully implemented into the R4F platform as a concrete co-simulation example by using the FMI and SSP interface standards (see [Subsection 5.1.3](#)), to predict more optimal P, I and D parameters for more stable and accurate simulation results. Besides, Neelamraju [90] found out and investigated standardization methods to contribute to the implementation of a co-simulation framework into a DT of a ship powertrain system for the maritime industry. In the meanwhile, he promoted the SSP standard as a standardized interface to represent, describe, keep the structure and requirements of the system, and also to easily exchange, integrate and co-simulate the corresponding models as FMUs. He proposed to use an ontology together with

the SSP standard to describe component requirements of the powertrain models, as well. Beside that, he noticed that the SSP has limitations such as lack of standardized model requirements, and limited interoperability with other external models between various software tools and platforms, which should be overcome to enable and enhance accuracy, interconnectivity and compatibility in a co-simulation framework.

3.3 Research Gaps and Contributions

In this subsection, the filled research gaps, scientific contribution and novelty of this dissertation project, which come as outputs of the comprehensive state-of-the-art investigation mentioned above, are exposed from the perspectives of AIP in DTs, DevOps automation and management for simulations in DTs, and co-simulation implementation into DTs.

First, regarding to the integration and processing of simulation assets in DTs, there are many research examples handling the use and adaptation of simulations in different DTs of different physical systems. However, no research finding is found so far, where a particular AIP methodology is designed and developed to effectively integrate and process simulation assets in a railway DT platform at open-source level by using interface standards and software tools. This dissertation project aims to fill the research gap between simulation assets and a railway DT platform by supplying such an AIP methodology as previously mentioned. (for more information about the proposed AIP methodology see [Subsection 4.2](#))

On the one hand, many research papers exist, which mainly deal with the automation and management of the DevOps process of physical entities in conjunction with DTs of these entities. On the other hand, no research work supplies a concrete and comprehensible methodology to automatically integrate, process simulation assets incl. co-simulations, and manage it in a suitable test environment, which is based on a railway DT platform and acts as a preliminary test station, at open-source level by utilizing DevOps solutions. This dissertation work provides such an automation and management methodology followed to automate and manage the AIP in the R4F Platform. (for more information about the proposed automation and management technique incl. the main steps see [Subsection 4.3](#))

Regarding to the co-simulation implementation into DTs, many research findings implement co-simulations into different DTs of different physical systems. However, there is no concrete research example proposing a co-simulation implementation methodology for the integration and processing of co-simulation assets centrally in a railway DT platform by utilizing open-source interface standards and software tools. Beside the centralized co-simulation implementation, there is not any distributed co-simulation technique, proposed to split co-simulations of different simulation models across multiple machines or workstations, which might represent railway DTs belonging to different stakeholders, at open-source level. This dissertation work supplies specific techniques helping to implement co-simulations into a railway DT platform, and to distribute these into several representative machines by using various open-source interface standards and tools under consideration of computation resource consumption and IP protection issues. (for more information about the co-simulation implementation methodology see [Subsection 4.4](#))

4 Methodology and System Architecture

This section aims to give valuable insights into the proposal methodology and system architecture, which help to fill the research gap between simulation assets and a railway DT framework such as the R4F Platform by defining the AIP task, performing, automating & managing it based on a workflow, and implementing co-simulations into a virtual test environment in a centralized and distributed fashion at open-source level. As previously mentioned in [Subsection 1.7](#), the environment is used for the research of this thesis and is the simplified and compatible version of the railway DT platform.

4.1 Task Definition

The AIP task is the core mission of this dissertation project, which is defined, described and concretized as one of the main parts of the methodology in this subsection, to fill the research gap between railway simulation assets and a railway DT platform as previously mentioned. The task definition helps to realize the AIP, thus answers the RQ1 (see [Subsection 1.4](#)) and belongs to the Research Cycle 1 of this dissertation work mentioned in [Subsection 1.7](#). It is also important to notice that the definition and concretization of this task are based on the ESDT-related state-of-the-art investigation of this work. This main task consists of two subtasks: Asset Integration and Asset Processing.

4.1.1 Asset Integration

The first part of the main task, which is applied to the R4F Platform for the Rail4Future project, is called Asset Integration. The Asset Integration part includes building raw railway simulation assets as adapted to the platform. The building process applied as it follows:

- **Simulation modeling:** In this sub-process, first, a simulation model of a physical system has to exist, which is the basics to build the system's DT by using the simulation modelling approach, and represents a use case such as the MBS virtual vehicle, RLT railway bridge, ML-based surrogate model and anti-slip traction behavior and vehicle speed control co-simulation model as concrete examples mentioned in [Subsection 5.1](#). In some cases, further development and modifications of the simulation model, supplied from the Asset Provider (see [Subsection 1.5](#)), might be necessary to clearly succeed the simulation

process incl. input parameterization and output generation. For example, for the anti-slip traction control co-simulation use case, the MBS vehicle model was needed to be extended by implementing and then connecting new force elements to newly created input channels, which are connected to a control model to control the traction of the virtual vehicle (for more information see Wurth [141]). To further shape and modify a simulation raw model, advanced simulation software tools with Graphical User Interface (GUI) are typically utilized (e.g., Simpack, MATLAB). Besides, input data are defined, configured, entered to the model, and finally the simulation of this model is tested with these data to imagine various scenarios and also to check if the simulation outputs are plausible. As a concrete example, the track arc radius and track superelevation were needed to be concretized and set in the MBS vehicle model to realize the curved and straight vehicle ride scenarios in a virtual environment (see [Subsection 5.1.2](#)).

- **Input & output registration:** In this second step, first, concrete input parameters, which can be directly entered by the users of the R4F Platform afterwards, are defined and registered to the simulation raw model to provide input parameterization. This might also help them to do scenario analyses through parameter studies in a DT-based simulation environment. After that, output channels are specified and then assigned to the model in order to extract simulation outputs, which can then be utilized by the users for result validation, data analysis and visualization purposes, from these channels, as well as to bind these output channels to other simulation models for co-simulation implementations. Lastly, concrete input channels are defined and registered, which are important to receive data from other external models through connections, and thus to manipulate the model for co-simulation purposes as demonstrated in the anti-slip co-simulation use case example (see [Subsection 5.1.3](#)).
- **Model containerization:** After that, the simulation model need to be containerized for more description, tool independency, platform compatibility and file portability. For this, open-source interface standards are highly preferred, which are mentioned in [Subsection 2.2.1](#) in detail. For example, based on the IPSE- and CSI-related state-of-the-art investigation of this work, the FMI standard, which is already supported by more than 250 tools [133], is highly promoted and thus utilized by many researchers to serve raw simulation models as a black-boxes, containing predefined and registered input parameters, input & output channels, for their successful adaptation to various DT environments as found out in this investigation. Figure 4-1 in [Subsection](#)

[4.2](#) shows the model containerization technique exactly by utilizing such an interface standard.

- **Asset application design, test and implementation:** As the last step, a simulation asset application, working in the background by utilizing a CLI-based interface instead of a GUI, is designed, developed, tested to verify its functionality, and then implemented into the R4F Platform to execute a simulation model after its successful containerization incl. input parameterization. A methodology, followed to design and develop such an asset application by utilizing standardized interfaces and the OOP technology (see [Section 2.2](#)) for the AIP task in the platform, can be found in Figure 4-3.

4.1.2 Asset Processing

The Asset Processing, which is the second part of the main task, is required to extensively examine, further improve and then deliver final simulation assets, which are previously built in the Asset Integration part, to the stakeholders of the Rail4Future project. The Asset Processing part is divided as it follows:

- **Code analysis:** First, source codes, which are written to build and develop the basics of railway simulation asset applications implemented into the R4F Platform, are analyzed to detect potential bugs, security vulnerabilities and code redundancies (e.g., exposed user credentials, unused software package imports). Typical code analysis examples can be found in the DAS-related state-of-the-art investigation of this work. According to this investigation, researchers tend to use static CAT for such purposes mentioned above to further improve the quality of their source codes used for simulations in DTs. The typical implementation of a code analysis process into a CI/CD pipeline, used for the automated AIP, can be found in [Subsection 4.3](#) (see the Analyze Codes step in Table 4-1).
- **Model and data validation:** After analyzing the source codes, railway simulation assets, consisting of the containerized simulation models and their belonging data, need to be validated to find out if these assets can work properly, are compliant with some specific standards (e.g., FMI compliance) and show name consistencies. For example, the FMU models, containing simulation algorithms of railway use cases, might possess corrupt binaries or syntax errors in their modelDescription.xml file, which can be automatically detected by using the FMU Compliance Checker tool [50]. This tool was also mentioned once in the IPSE-related state-of-the-art investigation of this work while speaking about the need for the FMI evaluation. Another concrete example is that the names of

input & output channels, mentioned first in the system description file of an SSP, and second, in the XML-based model description file of the corresponding FMU component, are checked by reading these files to determine the name consistency. The exemplary implementation of such typical validation processes into a CI/CD pipeline for their automation can be found in [Subsection 4.3](#) (see the Validate Interfaces and Validate FMU steps in Table 4-1).

- **Simulation test:** After the model and data validation step, the simulation of the containerized simulation models need to be tested in the R4F Platform to see if the model simulation really works, provides the desired, reasonable and stable system behavior with corresponding plausible input parameterization in the platform. For this, external input parameters are read, then sent to the calculation algorithm of the model, the algorithm is invoked, and finally the corresponding results, giving comprehensive insights into the model behavior, are yielded. [Subsection 4.2](#) shows exactly how the entire simulation test is performed by utilizing a simulation asset application.
- **Result validation:** After the blackbox-based models passed the simulation test in the R4F Platform, their results are then generated as validation curves or points, where the simulation results of the containerized models are visually presented as a comparison between these results and the previous ones achieved directly from the default simulation software tool. By this, scenario characteristics, system behavior and result consistencies are determined. This would then be used for predictive analyses, maintenance and conditional monitoring of a physical domain such as a holistic large-scale railway infrastructure system, mentioned in the Rail4Future project, in a DT-based framework. There are also several result validation examples, belonging to the railway use cases in [Section 5.1](#), as shown 2D visual diagrams in [Subsection 5.2](#). According to the state-of-the-art investigation of this work, several researchers already used the DT technology with simulations as a validation framework helping to gain useful insights into the analysis and inspection of the physical system in different industrial fields. In the Rail4Future project, the R4F Platform was presented as a virtual validation framework, which is based on a DT conception solution, for the railway industry [147, 108].
- **Asset delivery:** This final step aims to securely release the final version of the railway simulation assets to the Infrastructure Layer of the R4F Platform, which only authorized users utilize (see Kugu et. al. [30]). A similar step to this is typically implemented into CI/CD pipelines under the name of CD, aiming to continuously deliver or deploy software

applications in conjunction with version controlling of these applications as found out in the state-of-the-art investigation of this work. A concrete method to apply the asset delivery process into the CI/CD pipeline can be found in [Subsection 5.2](#) (see the Deliver Asset step in Table 4-1).

4.1.3 Role and Importance for the Rail4Future Project

In this subsection, the role and importance of the AIP task for the Rail4Future project are pointed out in terms of preparation of simulation asset applications to realize various railway use cases in the R4F Platform.

The main task helps to build, test, optimize and then release these asset applications to other stakeholders in the platform. By this, the simulation assets are ready to be used in terms of platform compatibility, reusability, comprehensibility, portability and traceability for predictive analysis, maintenance and conditional monitoring of particular subsystems (vehicle, track, turnout, bridge, tunnel), belonging to the holistic large-scale railway infrastructure system, in the DT platform. These are performed by the users of the platform, who are railway infrastructure managers and train operators from the railway sector. Thus, the AIP task is extremely important to apply all these simulation assets to the platform for problem diagnosis, predictive maintenance, conditional monitoring and intelligent decision making of the railway infrastructure system [108], which highlights the success of the Rail4Future project.

As previously mentioned, this dissertation work participates in the Rail4Future project as an Asset Integrator, who is responsible for the integration and processing of the simulation assets, incl. co-simulations, in the R4F Platform as well as for the automation and management of the AIP task by implementing exemplary use case specific simulation pipelines due to the feedback from other stakeholders. These pipelines are then used to execute the simulation of these models belonging to these assets, and finally deploy these into the visualization to the users in the platform. (see Zhou et al. [147])

4.2 Asset Integration and Processing Methodology

This subsection clearly exposes the entire basic methodology, which helps to realize the AIP task, mentioned in [Subsection 4.1](#), thus answers the RQ1 (see [Subsection 1.4](#)) and is applied in Research Cycle 1 of this dissertation work mentioned in [Subsection 1.7](#). This subsection shows and describes methods

and approaches, which are designed and developed based on the state-of-the-art investigation of this work (see [Section 3](#)) to apply the sub-processes of the AIP task mentioned in [Subsection 4.1.1](#) and [Subsection 4.1.2](#), and are necessary to build, test and then deliver simulation assets to other stakeholders in the R4F Platform. Thus, these methods and approaches are essential to create the proposal system architecture in this work.

As the first step of the Asset Integration part, a concrete simulation model needs to exist, which belongs to one or more railway subsystems, and represents a railway use case. For this, advanced simulation software tools are used to build and test the model in a virtual environment (e.g., Simpack for MBS vehicle modelling). In the building process, many different kinds of virtual components, representing their physical versions, are combined to each other, different calculation algorithms are used depending on what functionalities and results are demanded to achieve from the simulation testing. For the Rail4Future project, simulation models were provided from different stakeholders, who are simulation experts and play the Asset Provider role of the project, at the beginning. Then, these models are analyzed and tested using their default simulation software tools, which is necessary to comprehend the entire simulation process of these models incl. input parameterization and output generation for their integration into the R4F Platform. In the meanwhile, it is notable that the need for software licenses to build and test some of these models cannot be denied. This also shows a brief insight into the necessity of software license management in the platform. Besides, some of these models are extended in terms of functional modelling using their default tools again due to the needs coming from the end users of the platform. For example, the MBS railway vehicle model needs to be further modeled to control its anti-slip traction behavior and vehicle speed in a virtual environment, which is mentioned in [Subsection 5.1.3](#).

After creating, testing and optimizing the simulation models, input parameters, input & output channels need to be registered into these models in order the users to utilize these components in the platform for transparent input parameterization, co-simulations, result extraction for validations, data analyses and data visualizations afterwards. This is also essential for them to do predictive analyses, monitoring and issue tracking of the railway infrastructure system in the DT platform. For the registration process, several particular key data of these registered parameters and channels are defined according to the

FMI specification document [38] before packaging these models into FMU. These key data are:

- **Variable name:** The name of the parameter or channel, which is directly used in the models as input or output.
- **Start value:** The value, that the variable becomes at the beginning.
- **Reference value:** Identification number of the variable to address it in the FMU.
- **Data type:** A representative group of values, which the variable can become. These are the existing data types used for the FMI version 2.0:
 - *Integer* → A range of numbers without decimal points (e.g., 343),
 - *String* → A group of characters to save text (e.g., “my favorite text”),
 - *Real* → Floating-point numbers incl. decimal points (e.g., 0,012),
 - *Boolean* → Includes two representing data, which are *true* and *false*,
 - *Enumeration* → A group of values representing any particular attribute (e.g. *min* and *max* defining the value range of the variable)
- **Description:** A string indicating the meaning of the variable (optional).
- **Variability:** This shows when the variable can change its value. It can be one of the followings:
 - = “*constant*” → never changes,
 - = “*fixed*” → no changes after initialization,
 - = “*tunable*” → changes in the next external event (Model Exchange), changes in the next communication point (Co-Simulation),
 - = “*discrete*” → doesn’t change between external and internal events during simulation time, state, step (Model Exchange), or only differs at communication points (Co-Simulation),
 - = “*continuous*” → the variable may differ (Model Exchange) or comes from differential (Co-Simulation).
- **Causality:** This means the type of the variable, which can be one of these (only the types used for this work are given, for further information see [38]):
 - = “*input*” → is an input channel providing values as inputs in the models,
 - = “*output*” → is an output channel providing values as outputs resulted from the model simulation,

- = “*parameter*“ → identifies input parameter, set independently and cannot be used for connections.
- **Initial:** This shows how to initialize the variable, which can be one of the following values:
 - = “*exact*“ → initialized with a *start* value,
 - = “*approx*“ → an iteration variable of an algebraic loop starting with a *start* value.
 - = “*calculated*“ → calculated from other variables.

After defining and registering all the parameters, input & output channels, the models are containerized into FMU using the FMI standard, because this standard is open-source and provides tool independence, platform compatibility, model description and file portability. Besides, the FMI is so popular among researchers and engineers, that it is supported by relatively many tools, which are designed and developed by using open-source and commercial software technologies incl. OOP languages. This also makes the standard flexible and attractive for adaptation of different simulation models to any virtual environment of different industries, which is already proved by the state-of-the-art investigation of this thesis (see [Section 3](#)). After the FMU containerization process, one or more FMUs are packaged into SSP, which is open-source, provides descriptions (incl. system structure, parameters), an additional containerization level and co-simulation abilities, making connection of two or more FMU-containerized models possible for co-simulation cases, in addition to the FMI standard. Therefore, the SSP is especially preferred for co-simulation of two or more FMU models, and transferring the entire co-simulation incl. the FMU components as a file. Although less amount of software tools support the SSP compared to the FMI, the SSP gains more interests in recent years for co-simulation implementation tasks in different industries incl. the railway sector, which is also proved by the CSI-related state-of-the-art investigation of this work (see [Subsection 3.2.2.3](#)). For the Rail4Future project, the FMI for Co-Simulation is preferred, which is suitable for the integration of all the simulation models into the R4F Platform without needing any extra simulation environment. As previously mentioned, there are several tools to package one or more models into FMI and/or SSP, which are carefully chosen according to the default simulation tool, OS-compatibility, FMI & SSP version numbers, their compatibility to each other, and programming language of the model (see [133] and [134]). Besides, it is important to tend to stay at open-source level to avoid additional fees for software licenses in future, which

increase in case of more computers due to the amount of stakeholders by time. These licenses might be necessary to start the model simulation as previously mentioned.

Figure 4-1 exactly shows the model standardization approach, which is proposed by Kugu et al. [76] and indicates the FMI- and SSP-based containerization technique used for the model integration in the R4F Platform. On the left side of this figure, there are all the raw simulation models (e.g., RLT steel bridge) coming from the Asset Layer of the R4F Platform (see Figure 1-2), where the Asset Provider keeps these models. Then, these models are packaged into an FMU file (*.fmu), which contains the components as it follows: (for their detailed description see [Appendix Subsection A.2.1](#))

- **modelDescription.xml file:** It includes the concrete description of the containerized simulation model incl. model metadata, input & output connectors and parameters,
- **binaries folder:** This folder contains OS-dependent binary(-ies), which is (are) important to ensure the platform compatibility of the model,
- **sources folder (optional):** This is an alternative folder, containing C resources for FMU compilation and linking purposes (e.g., the FMU of the RLT calculation algorithm, belonging to the RLT bridge use case (see [Subsection 5.1.1](#)), contains such a folder),
- **resources folder (optional):** This alternative directory contains additional input data, which might be needed in order the FMU to read those for its initialization (e.g., tabular input data, the input parameterization of which is not supported by the FMI 2.0 version because of their high data complexity as found out in this work, are read by the FMU of the RLT calculation algorithm for the RLT bridge use case).

After the FMU packaging, the FMU(s) is(are) containerized into SSP, which contains the components as it follows (for their detailed description see [Appendix Subsection A.2.2](#)):

- **FMU component(s):** containing the simulation model(s),
- **SSD file:** describing the entire system structure of the simulation system incl. system metadata, input & output connectors and connections (for co-simulation cases),
- **SSV file(s):** describing user input parameters of the FMU component(s) incl. parameter names, values and data types,

- **extra folder (optional):** An alternative folder, considered to keep additional input data sets and raw simulation results, utilized for result validations, as files and folders [76].

Figure 4-1: Model standardization approach. [76]

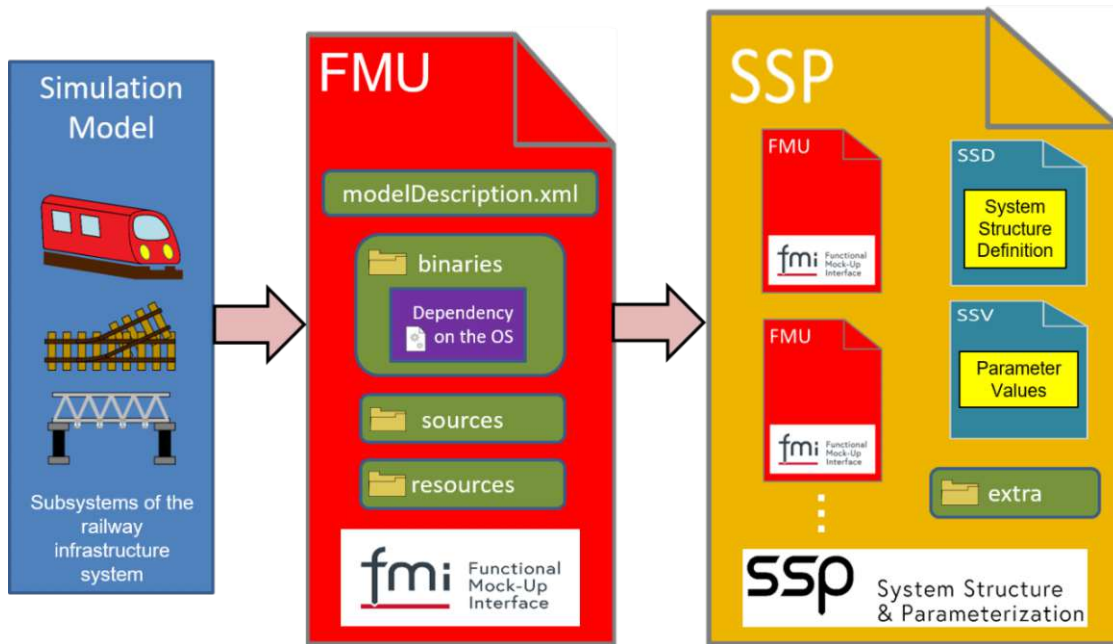


Figure 4-2 briefly indicates the manual code processing technique to prepare the raw model, written in a Python code, for its FMU containerization using a shell command, which is based on the *PythonFMU* Python library [55]. This library is open-source and qualified to containerize Python codes into FMU, therefore it is preferred for the Rail4Future project. First, all the functions of the code are directly migrated into a newly created Python class (first class). Then the remaining code lines, including the use of the functions and different variables for a calculation, are moved into another Python class, written under the first class. Besides, inputs & outputs of the model are registered in the code as the last step.

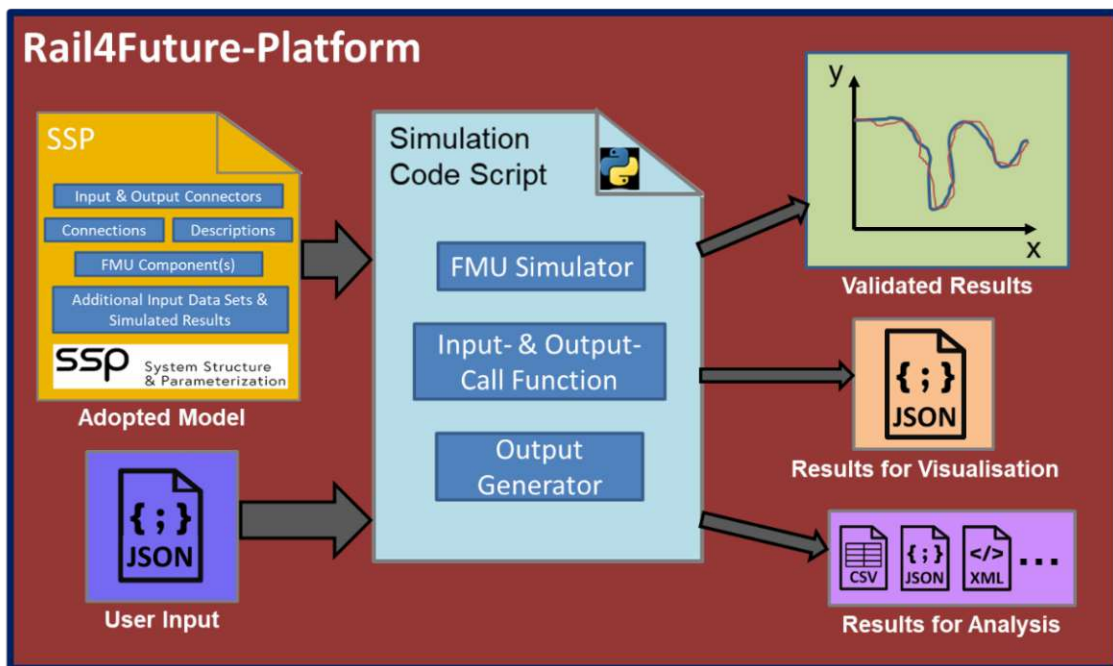
Figure 4-2: Manual code processing for model integration with FMI. [29]



After containerizing all the simulation models and data into FMI and SSP, an asset application needs to be designed and developed, which helps to build and test the simulation process in a representative simplistic and compatible machine of the R4F Platform such as the virtual test environment, which is used for the research process of this dissertation work mentioned in [Subsection 1.7](#). For this, a model simulation approach is considered and followed, which is proposed by Kugu et al. [76] (see Figure 4-3). In this approach, first, there is the adopted model, which is successfully containerized into FMI and SSP. The SSP contains input & output connectors (channels), which are connected to each other and therefore connections come out in case of co-simulation, descriptions, describing the system structure (.SSD file) and parameters (.SSV file), and the

FMU component(s), representing the FMI-containerized version of the raw simulation model(s). Besides, additional input data sets, needed for input parameterization, and simulated results for validation purposes are added to the SSP file. After finishing the SSP containerization, an input JSON file, which is based on the R4F standard (for an example see Appendix 5 in [Appendix Subsection A.1](#)), is used to enter user inputs, read and comprehend all the input parameters of the model(s) incl. their descriptions. Then, a simulation code script is written in Python programming language, based on the OOP technology, to execute the FMI-based SSP simulation with input extraction from the JSON file, and output generation after the simulation process. For this execution, the script has an FMU simulator, which uses the *fmpy* Python library [49] to start the FMU(s) as a black box. For the input extraction from the JSON, the script has an input call function, which utilizes the *json* Python library [106] in addition to the *fmpy*. To extract outputs from the simulation for the output generation, an output call function is applied to the script. For this, only the *fmpy* is used. Lastly, the script includes an output generator, helping to generate validated results, outputs for visualization and data analysis. The output generator uses the *matplotlib* Python library [83] to publish the validation results as image(s), and *json* to create JSON output, based on the R4F standard, for result visualization to the users of the platform. Additionally, there are other Python libraries, which can be utilized to generate other kinds of outputs (e.g., *csv* Python library [105] for CSV file generation for tabular data analysis).

Figure 4-3: Model simulation approach. [76]



4.3 Automation and Management

This subsection presents the entire methodology to build a system architecture for automation and management of the AIP task in the R4F Platform, which helps to answer the RQ2 regarding to striving for full-automation (see [Subsection 1.4](#)) and is applied in Research Cycle 2 of this dissertation project mentioned in [Subsection 1.7](#). First, an overview of the main approach, followed for the automation and management, is shown and described. Then, the design and implementation of a CI/CD pipeline are implied, which provides a CI/CD process, mentioned in [Subsection 2.2.5](#), for workflow-based DevOps automation of sub-processes by utilizing different software tools and technologies in conjunction such as VMs, software containers, VCS and CAT. These sub-processes are defined to continuously, effectively build, test and finally deliver simulation asset applications, belonging to different railway use cases (see [Subsection 5.1](#)), to other stakeholders, who can then directly visualize these and give the control of the asset applications to the end users of the platform. Besides, a structural overview of the file system of a typical asset application, uploaded into a software repository of the VCS belonging to the Asset Integrator, is briefly shown and described. Lastly, it is presented what is documented as a result of the automation and management.

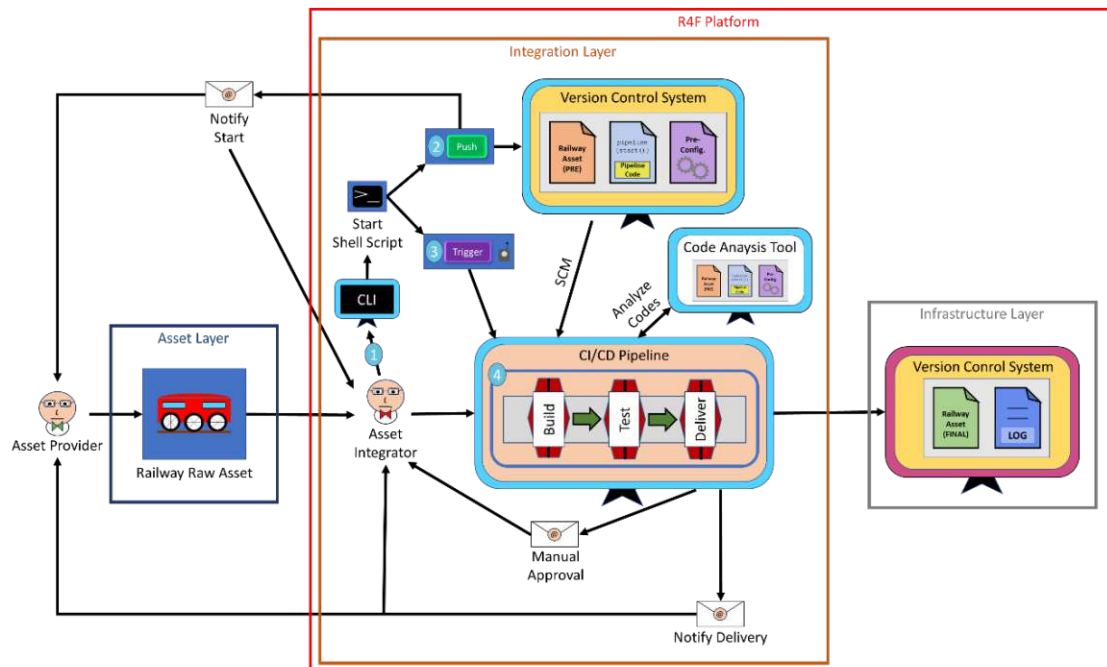
4.3.1 Main Approach

To provide an ideal approach for the automation and management of the AIP task, a series of criteria are taken into account such as time-, cost-efficiency, practical orientation, function reliability, quality control, version control, team collaboration and workflow definition. According to the research findings, coming from the DAS-related state-of-the-art investigation of this dissertation work, researchers proposed various frameworks, approaches and methods to automate the DevOps process of a DT or physical system incl. building, test, delivery/deployment of simulations by utilizing potential key technologies, tools and applications under consideration of such criteria. However, due to different industry fields, various DTs, simulation models, software tools and applications are involved in those proposals meaning requirement differences even if those are inspiring to find out an optimal approach for the DevOps automation task. To propose such an approach for the AIP automation and management in the R4F Platform at open-source level, potential key technologies such as the CI/CD pipeline technology, VCS, CAT, VM and software containers, mentioned in [Subsection 2.2.5](#) and also in the research findings, are picked, tested, integrated with each other and other technologies such as the OOP, FMI/SSP standards, OSP software, other open-source tools (e.g., CLI, shell scripting technologies for command executions). All these selected technologies are then applied to create, further develop and optimize the virtual test environment, representing a simplistic and compatible version of the R4F Platform as mentioned in [Subsection 1.7](#), to fulfill the AIP automation task for the research of this work. Figure 4-4 simply indicates such an approach, proposed by Kugu et al. [77] and followed to automate and manage the integration and testing of simulation assets in the R4F Platform at open-source level for the Rail4Future project. Besides, this figure addresses some of the layers, which are defined by Zhou et al. [147] to conceptualize the platform, to the approach to show the significant relevance of this approach to the Rai4Future project.

As the first step of the AIP automation and management, shown in this figure, the Asset Provider collects all the railway raw simulation assets in the Asset Layer. Then, the Asset Integrator receives these assets from the Asset Layer in order to automatically integrate and process these in the Integration Layer of the platform. To initialize the AIP, a shell script is used to start the engine on the CLI, which helps the Asset Integrator to push all the relevant files of the assets (see [Subsection 4.3.3](#)) into their VCS, and then pull the trigger of the CI/CD

pipeline. These files include the pre-assets (pre version of the assets), pipeline code script, defining the automation workflow of the pipeline, and configs (configuration files), containing necessary configurations (e.g., for the right connection of the pipeline server to the CAT). In the meanwhile, the Asset Integrator and Asset Provider get automatically notified about the pipeline start via email using an open-source mail transfer agent software in the platform. In this email, only the Asset Integrator receives additional four links, which belong to the pipeline, CAT, VCS tool and pipeline console output, due to IP protection reasons regarding to the pipeline automation methodology. This also assists them to easily manage the entire automated integration and processing of these assets. After that, the pipeline starts with three main processes: Build, Test and Deliver. The names of these processes come from the CI/CD term, mentioned in [Subsection 2.2.5](#), which cover building, testing and delivering of software applications. In the case of this work, simulation asset applications come into focus, which are automatically built, tested and finally delivered by the CI/CD pipeline within the automation and management approach, instead of software applications (for more information about further description and concretization of these main processes see the next subsection). Within the Test process, the pipeline automatically sends an email to the Asset Integrator. This email includes the manual approval message and the link of the pipeline incl. IP address and port number, where they can easily access the pipeline server. The Asset Integrator needs to manually approve the asset delivery before releasing these assets to the VCS of the Infrastructure Layer. In this layer, the Asset Integrator monitors simulation pipelines of the Function Layer, and run simulations by executing the pipelines in the platform. After they manually approve it, the Deliver process come out into being, namely the final assets are smoothly delivered to the Infrastructure Layer of the platform and then all the stakeholders automatically receive an email indicating the information about the successful asset delivery.

Figure 4-4: Automation and management approach. [77]



Main Procedure Followed	
1	Start the Shell Script
2	Push Pre-Assets and Notification
3	Remotely Trigger Pipeline
4	Build → Test → Deliver

4.3.2 Pipeline Design and Implementation

This subsection provides an insightful overview of the automation workflow, which includes main processes and their sub-processes. This workflow is designed and developed using the CI/CD pipeline technology, and can be controlled editing a pipeline code script, which is also uploaded into the VCS of the Asset Integrator as previously mentioned.

Table 4-1 shows and describes the sub-processes, which come from the main processes based on the *Build*→*Test*→*Deliver* principle. First, the *Build* step helps to prepare the simulation assets and pipeline environment by connecting the pipeline to the VCS through the use of the Source Code Management (SCM) system, updating software tools, packages and libraries, and containerizing these assets into FMU. Second, the *Test* step aims to test the entire simulation asset application by validating its FMU containers, checking

name consistencies between its different files, analyzing its source codes, gathering its FMU metadata for information checks, updating its SSP, testing its FMI-based SSP simulation, generating its simulation results and finally establishing a manual approval sub-process for the Asset Integrator's trusty review of the asset application. Lastly, the *Deliver* step assists to smoothly release the asset application to the Infrastructure Layer's VCS in the platform by invoking some particular execution commands based on *Git* [45].

Table 4-1: Sub-processes of the CI/CD pipeline. [77]

Main Processes	Sub-Processes	Description
Build	1) Checkout SCM	The CI/CD pipeline is directly connected to the Asset Integrator's VCS through the use of the SCM system, which the pipeline server supports.
	2) Update Software	Software tools, packages and libraries are auto-updated using software package managers (e.g., yum) in conjunction with execution commands.
	3) Build FMU	Simulation models are auto-packaged into FMUs using their default simulation tools of the models by command executions instead of opening any GUI of these tools.

Test	4) Validate Interfaces	The parameter, FMU file, FMU component, input & output connector names of the input JSON and newly created FMU files are compared to these of the example SSP file for interface validation in terms of name consistencies by only executing a Python code, which uses the <i>BeautifulSoup</i> [12], <i>zipfile</i> [151], <i>os</i> [96] and <i>json</i> [106] Python packages.
	5) Analyze Codes	The source codes of the asset application are analyzed using the CAT to find out security vulnerabilities, code redundancies, errors and bugs.
	6) Validate FMU	It statically checks the FMI compliance of the FMU files in terms of structure and properties using the <i>fmpy</i> Python package.
	7) Gather Info FMU	FMU metadata (e.g., FMI version, model name) are automatically extracted from the FMU files using the <i>fmpy</i> Python package.
	8) Update SSP	An example SSP, once created using the Model.CONNECT tool, is updated by auto-uploading the newly created FMUs into it. For this, some typical execution commands are invoked by the pipeline.

	9) Simulate SSP	The FMI-based SSP simulation of the assets is automatically started. During the simulation process, user input parameters are read from an input JSON file, then directly entered to the simulation and finally simulation outputs are extracted after FMU execution.
	10) Generate Results	After finishing the simulation, the simulation results are automatically published in different formats (e.g., CSV-tables, validation curves or points, JSON format) for validation, visualization and data analysis purposes.
	11) Manual Approval	The Asset Integrator checks all the simulation results, code analysis reports and pipeline console outputs. If all is in order with the entire asset application on their side, they manually approve the asset delivery to the Infrastructure Layer's VCS in the platform.
Deliver	12) Deliver Asset	The assets are auto-delivered seamlessly to the Infrastructure Layer's VCS using the <i>Git</i> command [45].

4.3.3 File System Structure in Version Control

In Table 4-2, an overview of the entire file system, which belongs to the released asset application, is clearly shown and described. This includes various files and folders, which are used by Kugu et al. [77] for the automated integration and delivery test of the simulation assets in the R4F Platform.

Table 4-2: File system structure of the asset application released in the R4F Platform.

Folder	File	Description
r4f-asset-application	Code analysis configuration file	The file helping to configure the code analysis process. This includes typical properties such as project key (unique key for the project), project name and/or project version.
	Rail4Future use case model 1 (r4f_uc_model1.fmu)	Containerized version of the raw models, which are containerized with the FMI standard as FMU files. These include components such as an XML-based model description file, binaries and others (for more information see Appendix Subsection A.2.1).
	Rail4Future use case model 2 (r4f_uc_model2.fmu)	
...		

	<p>Example SSP file (r4f_uc_example.ssp)</p>	<p>The container, which consists of a system structure file (SSD), one or more FMUs and their parameter description files (SSV). Optionally, the SSP file can include other files (for more information see Appendix Subsection A.2.2). For the Rail4Future project, the commercial Model.CONNECT tool is used to test the co-simulation of these FMUs as connected to each other, and then create this file one time as an example.</p>
	<p>Simulation code script</p>	<p>The key file, building the main interface of the entire asset application. This is executed to start the FMI-based model simulation incl. input parameterization and output generation. For the Rail4Future project, the code script is written in Python programming language.</p>
	<p>Pipeline code script</p>	<p>The code shows and describes the entire automation workflow, which is implemented into a CI/CD pipeline used for the automation and management of the asset integration and processing task.</p>

r4f-asset-application/ input_data	Rail4Future use case input parameter file (r4f_uc_input_params.json)	The main input file, which is human-readable and includes input parameters entered by the users of the R4F Platform.
	Additional input files, if necessary	The additional input files, which have a relatively complex and large data structure, and are necessary to be given to the model to start its simulation in the platform.
r4f-asset-application/ output_data	Default simulation results in CSV format (r4f_uc_results_default.csv)	The outputs, coming from the default simulation tool of the model (e.g., MATLAB) as a CSV table, used to compare these with the simulation results, generated from the FMI-based model simulation in the R4F Platform, for validation purposes.
	FMI-based simulation results in CSV format (r4f_uc_results.csv)	The outputs, directly coming from the FMI-based model simulation in the platform as a CSV table, used for data analysis purposes.
	FMI-based simulation results in JSON format (r4f_uc_results.json)	The simulation results, converted from the CSV results into JSON for data visualization to the users of the platform.

	Validation results as picture(s)	The outputs, which come from the comparison between the FMI-based and default simulation results, and are visually shown as curves or points.
r4f-asset-application/ raw_models	Rail4Future use case raw model 1 (e.g., r4f_uc_model1.spck) Rail4Future use case raw model 2 (e.g., r4f_uc_model2.slx) ...	The raw models, which are pre-built, further developed and tested in their default simulation tool. After uploading these into the platform, they are automatically packed into FMUs by the pipeline execution in the platform using the modules and libraries of the default tools (see Kugu et al. [77]).
r4f-asset-application/ software_packages	Additional software packages, if necessary	External software packages, which might be necessary to perform some automation tasks (e.g., FMI Kit for Simulink [40] software package to auto-package a Simulink model into an FMU).

4.3.4 Documentation of Results

Table 4-3 indicates the outputs, coming from the code analysis, FMI-based simulation and pipeline console, and documented as a result of the automated asset integration and processing performed in the R4F Platform for the Rail4Future project.

Table 4-3: Result documentation of the automated AIP.

Result Documentation	Description
1) Code analysis results	The documentation, concretely indicating and describing potential vulnerabilities, errors and bugs resulted from the code analysis process in consideration of key aspects such as maintainability, redundancy and security.
2) Simulation results	The outputs, generated from the model simulation process for visualization, data analysis and validation purposes.
3) Pipeline console outputs	A set of command outputs, which are resulted from the execution of the entire automation workflow, implemented into the CI/CD pipeline, in the pipeline console.

4.4 Co-Simulation Implementation

In this subsection, the proposed co-simulation implementation methodology is shown, which is designed and developed based on the AIP methodology, system architecture and the CSI-related state-of-the-art investigation of this work mentioned in [Subsection 4.2](#) and [Subsection 3.2.2.3](#). This methodology helps to integrate and process railway co-simulation assets (for a concrete co-simulation use case example see [Subsection 5.1.3](#)) into a railway DT platform at open-source level, thus applied in Research Cycle 3 of this work (see [Subsection 1.7](#)), and answer the RQ1 regarding to the realization of the AIP for co-simulation cases as mentioned in [Subsection 1.4](#). Two types of co-simulation implementation are succeeded and thus mentioned in this work: 1) Centralized co-simulation implementation; 2) Distributed co-simulation implementation. The first one was already demonstrated in the R4F Platform to integrate and process the simulation asset of the mentioned co-simulation use case example by utilizing the open-source FMI/SSP interface standards and OOP technology (see Kugu et al. [77]). There are already several co-simulation research examples from the state-of-the-art investigation, where researchers propose to use the FMI for containerization of simulation models, co-simulated in a DT-

based framework, and then the SSP for further containerization and description of the entire co-simulation model. The second implementation type refers to the distribution of simulation processes of each model across different machines/workstations, which is completed as an alternative solution to the centralized implementation type by utilizing the open-source FMI interface standard, VM and OSP software technologies. Besides, one goal of the distributed implementation is to answer the RQ3, mentioned in [Subsection 1.4](#), by striving for full IP protection of simulation models while sharing their simulation processes with other machines/workstations resp. other stakeholders. Several research examples also exist according to the state-of-the-art investigation, where researchers proposed to use the FMI standard, OSP application and also the DCP standard to split simulation processes into different subsystems, which might be a DT of a physical system and be installed in a VM, in terms of resource management, platform compatibility, interoperability and IP protection of models.

Figure 4-5 shows an overview of the approach, proposed and followed to integrate and process co-simulations centrally in the R4F Platform at open-source level by utilizing the FMI/SSP interface standards. This approach is actually an extended version of the AIP methodology, considered for co-simulation cases. The core difference of this approach lies on the SSP container, which contains a railway co-simulation model and consists of two or more FMU components, their input & output connectors and connections for linking of these connectors to each other. All these connectors and connections are established in a simple SSD file of the SSP. Besides, the SSP contains SSV files of these FMUs to keep their input parameters in descriptive files. Then, simulation results, which come from the co-simulation executed in the corresponding default simulation tools, additionally put into the SSP to keep these for later validation of the FMI-based SSP simulation results in the container. Lastly about the SSP, additional inputs, which contain highly complex data structures, might be put into the SSP, if these are needed for further input parameterization of the co-simulation process in the platform. Another difference of this extended approach is that the input JSON possesses two or more sections, which include use case specific input parameters of the two or more FMUs. It is also important to notice that the simulation code script needed to be further adapted to specifically work with SSP-based co-simulations instead of single simulations. For this, the *fmpy* Python library [49] had to be

further analyzed to find out its SSP-specific source codes utilized for the code development process.

Figure 4-5: An overview of the centralized co-simulation implementation technique based on FMI and SSP.

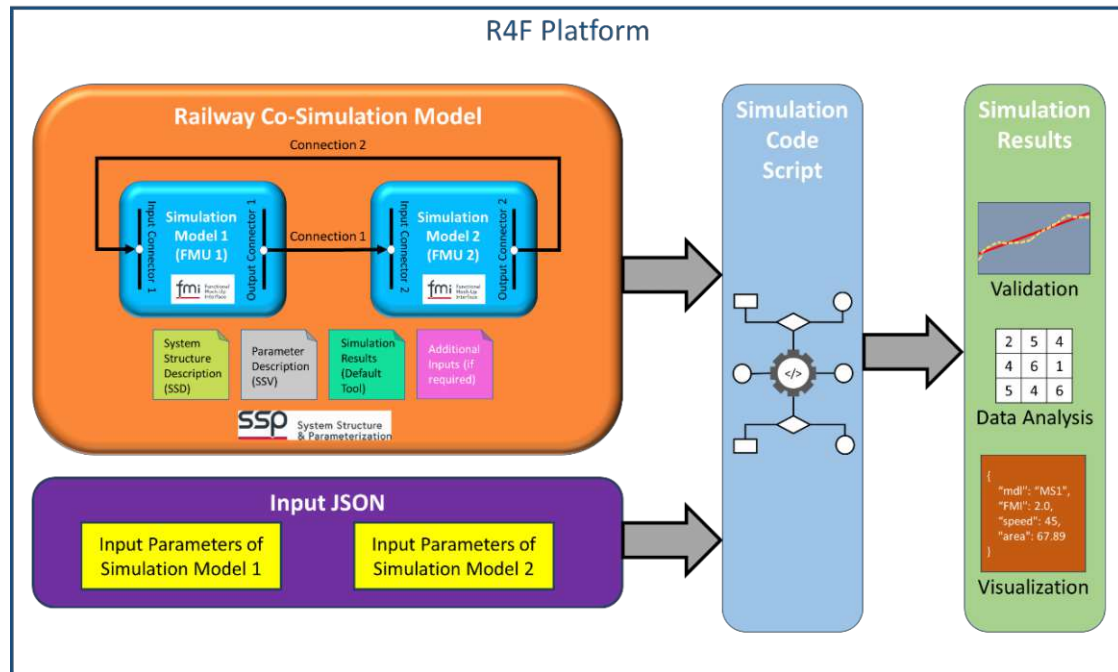
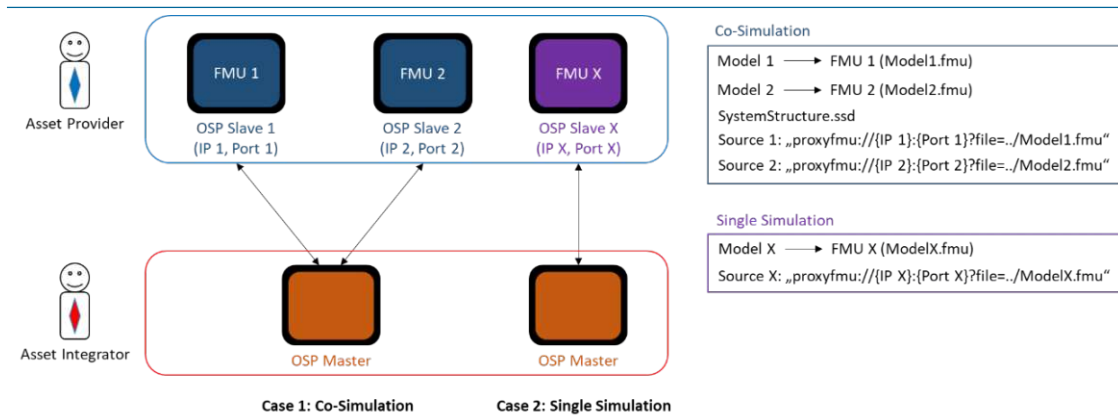


Figure 4-6 indicates an overview of a simple approach, which is similar to the DCP implementation technique (see Figure 2-1), proposed and followed to distribute railway co-simulations and single simulation into different machines using the open-source OSP software with the master-slave principle. In this figure, it is clearly shown and described how the Asset Integrator succeeds to distribute the co-simulation process from a railway DT platform into different machines using the OSP software (see [Subsection 2.2.2](#)). As the first step, the Asset Integrator prepares the FMUs and system description files of simulation models. These system description files include typical metadata of these models such as XML attributes (e.g., XML version), system name and description, source name(s), unit definitions, input & output connectors, parameters and connections (in case of co-simulation). The source names are entered with the IP address and port number of a remote machine (OSP Slave), which belong to the Asset Provider and where an FMU is instantiated from the

main machine (OSP Master), as started with the “**proxyfmu://**” string and ended with the file path of the FMU, that is necessary in order the OSP master to find the OSP slave as a target machine, where to auto-spawn the FMU instance. In case of the Rail4Future project, only the OSP Master, owned by the Asset Integrator, is put into the R4F Platform, from where the OSP Slave owned by the Asset Provider is kept according to the R4F landscape (see Figure 1-2). Second, ports are opened in the OSP slaves, where the distributed simulation processes arrive after starting the co-simulation process from the OSP master, by using the *proxy-fmu* application [54]. Lastly, they start the co-simulation or single simulation using the *cosim* application in the CLI [46] of the OSP master. In this case, they simply execute a command by entering the subcommand (*run* for co-simulation and *run-single* for single simulation), file path of the system structure file, and alternatively the output file path, where they receive the results in tabular format (e.g., CSV tables) in the OSP master.

Figure 4-6: An overview of the OSP-based distributed co-simulation technique.



5 Validation

This section illustrates the entire validation of the CI/CD pipeline and system architecture of the virtual test environment, representing a simplistic and compatible version of the R4F Platform, by presenting motivational railway use case examples at the first stage, which are key representatives for the evaluation process. Then, the FMI/SSP container version of these use cases is executed by the pipeline for its simulation, and after that their results are extracted for its comparison with the outputs of the raw model simulation, which was performed by utilizing default simulation tools of the use case specific raw models. By this, system behavior of the railway subsystems, belonging to the use cases, is comprehensively checked, result consistencies and stabilities are evaluated, which helps to judge the compatibility, reliability and fidelity of the pipeline and system architecture. It is also undeniably noticeable that the entire methodology and system architecture incl. proposed methods and approaches significantly helps to shape and improve the simulation workflow of the real-world use case examples in the R4F Platform by utilizing the key technologies (see [Subsection 2.2](#)) under consideration of open-source level tendency, software license management, quality assurance, version control, team collaboration, automation, interoperation, description transparency, software installation and configuration. Besides, this section provides discussions incl. remarkable benefits, drawbacks, challenges and limitations encountered during the AIP, its automation & management, and centralized & distributed co-simulation implementation.

5.1 Simulation Use Case Examples

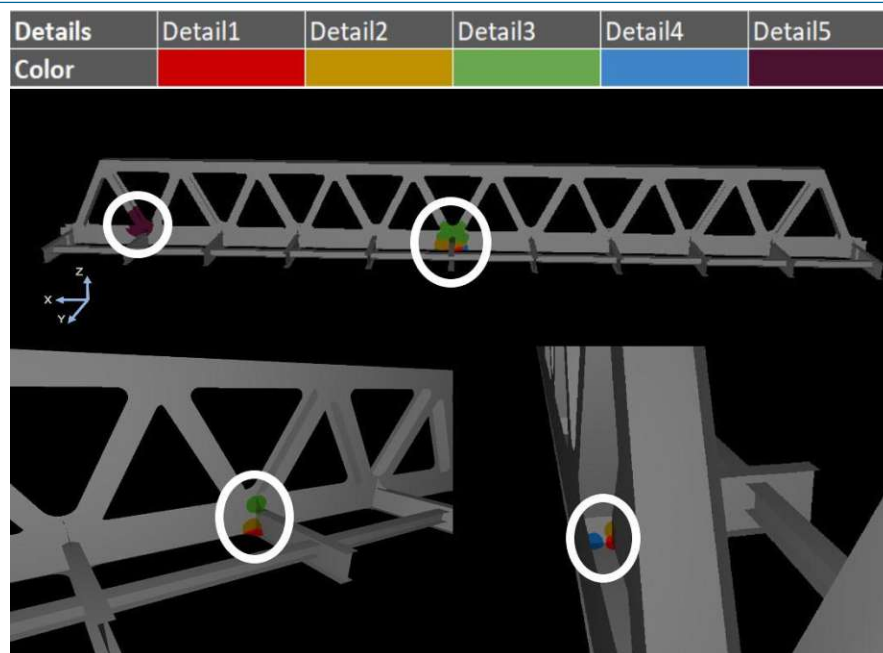
5.1.1 Residual Life Time (RLT) Calculation of a Railway Steel Bridge

In this subsection, a bridge use case is shown and described, which helps to calculate the residual life-time and damage sum of a railway steel bridge, and is successfully integrated and processed in the R4F Platform by following the proposed model standardization and simulation approaches in Figure 4-1 and 4-3 (see Kugu et al. [76]). In sum there are four bridges handled for the Rail4Future project and therefore mentioned in this dissertation work: dummy

bridge, ÖBB Eschenau bridge, ÖBB Mürzbrücke and Schellhamnergasse bridge (from Wiener Linien). The use case was provided from AIT as a Python code, containing RLT calculation algorithms, and CSV input files. Outputs are generated in another CSV file. For the Rail4Future project, the results of the use case are generated as a JSON file for human reading, and pictures for visualization of the validation points, resulted from the RLT calculation (see [Subsection 5.2.1](#)).

In Figure 5-1, first, the dummy bridge is introduced with its belonging detail points, as visualized by VRVis. Then, the other three real bridges are shown, which are visually obtained from the 3D Viewer app [1]. The inputs of the use case are detail positions (X, Y and Z coordinates), detail category, influence lines, frequencies of individual trains, axle loads and distances of the individual train configurations. The generated outputs are given as residual life time in years, and annual damage sum in 1/year.

Figure 5-1: Bridges used for the RLT calculation. [28, 76]



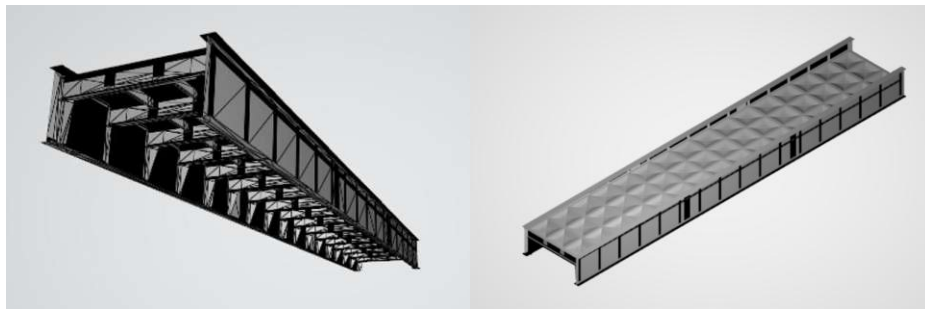
Dummy Bridge visualized by VRVis



ÖBB Eschenau Bridge



ÖBB Mürzbrücke



Schellhamnergasse Bridge from Wiener Linien

Exceptionally, there are other input types, which are specific for the real bridges. For example, the Schellhamnergasse bridge can be configured with a special boolean input variable, which activates the simplified fatigue damage curve (Wöhler curve) and therefore implemented into the Python code of this bridge. Besides, there are two types of influence lines, which are given as CSV inputs for the RLT calculation of Mürzbrücke and Eschenau bridges: initial (uncalibrated) and calibrated. Lastly, the load model, containing the frequencies of individual trains, is separately given as a CSV input for the RLT calculation of Eschenau and Schellhamnergasse bridges because of its variety.

To integrate the use case into the R4F Platform, the Python code is manually processed in order to adapt it for its FMU packaging by using the *PythonFMU* Python package [55]. For this, the manual code processing approach, shown in Figure 4-2, was followed. First, two Python classes are created. Then, all the Python functions, used for the RLT calculation, are moved into the first class,

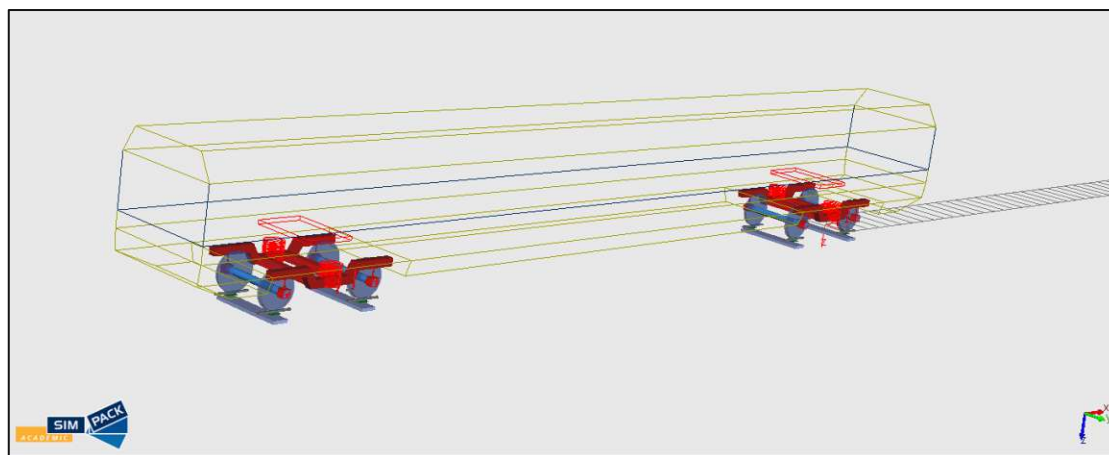
from which these functions are invoked for the entire simulation. After that, the remaining code parts other than the functions are migrated into the second class, where the entire simulation occurs. Lastly, necessary input and output channels are implemented into the second class in the code to give correspondent inputs and then get relevant outputs in the platform. As input channels, simply the names of the input & output folders, containing all the CSV input files, output files and pictures, are registered as strings due to the high data complexity caused by all the CSV inputs especially the influence lines. Otherwise, the exceptional inputs, mentioned above, are given as boolean and strings for the RLT calculation of the real bridges.

After finishing the manual code processing, the Python code is simply packaged into an FMU file, which includes two OS-dependent binary files (for Linux & Windows), an XML-based model description file, the Python code and C source codes, with the version of FMI 2.0 for Co-Simulation. Only for the dummy bridge use case, the one FMU is containerized into SSP using the commercial Model.CONNECT tool provided from AVL List GmbH. The reason for this is that the SSP standard is generally proposed to be used by Kugu et al. [76] for model integration into the R4F Platform. They also mentioned the successful integration of the dummy bridge into the platform with FMI and SSP as one of the first use case examples in their work. The SSP file contains an XML-based SSD, including system structure metadata (e.g., SSP version, system name, FMU component), input & output connectors, and the FMU file (for more information about the components see [Appendix Subsection A.2.2](#)).

5.1.2 Multibody Simulation (MBS) of a Railway Vehicle

This use case simply demonstrates the train ride on a track. For the Rail4Future project, an MBS virtual vehicle model, which is based on Manchester Benchmark [66], was previously designed and developed by Virtual Vehicle Research GmbH using the commercial Simpack tool from Dassault Systèmes, is seamlessly integrated and processed in the R4F Platform by following the proposed model standardization and simulation approaches in Figure 4-1 and 4-3 (see Kugu et al. [76]). Figure 5-2 shows a representative image of this vehicle, which has two bogies, and each bogie has two wheelsets.

Figure 5-2: MBS railway vehicle modelled in Simpack.



In Figure 5-3, the input & output definition and integration of this virtual vehicle with FMI into the platform is described and demonstrated. First, concrete inputs and outputs of the vehicle simulation need to be defined in order to reach the goal of the simulation, which then helps to calculate the life time of the vehicle in the platform. For this, there are two kinds of inputs: model and scenario parameters. Model parameters are immutable parameters, which characterize the model itself and were previously implemented into the model (e.g., geometrical length, simulation solver). On the other hand, scenario parameters are specialized to demonstrate different scenarios, therefore they are called so and can be changed spontaneously. For the Rail4Future project, vehicle speed (desired), track arc radius, superelevation, additional mass (passengers+luggage) and friction coefficient are specified as scenario parameters. By changing these, real-life scenarios such as straight/curved train rides, full/free wagons and summer/winter conditions can be imagined in the vehicle simulation. With the help of ViV, all the values of these parameters are specified in Table 5-1. In addition to these scenario parameters, desired track excitation was configured directly in the model by using default values, which were previously defined in Simpack (ORE high and ORE low). In the implementation of an ML-based surrogate model into the MBS vehicle, these desired track excitations can be given as CSV tables, which is mentioned in [Subsection 5.1.4](#) in details. After defining inputs, output generation plays a crucial part in result analysis and further use for the life time calculation of the MBS vehicle. For this, first, the vertical deflection of primary springs, implemented into four corners of each bogie, is generated as output. Besides, the actual vehicle speed, track position and actual left/right vertical track

excitations are set as outputs in the model. After testing the vehicle simulation with time integration, and registering necessary input and output channels to the model in Simpack, the FMI standard was applied to the model for its seamless integration into the platform, so that the model is simply containerized into an FMU file including a model description file and an OS-dependent binary file.

Figure 5-3: Input & output definition and integration of the MBS vehicle with FMI.

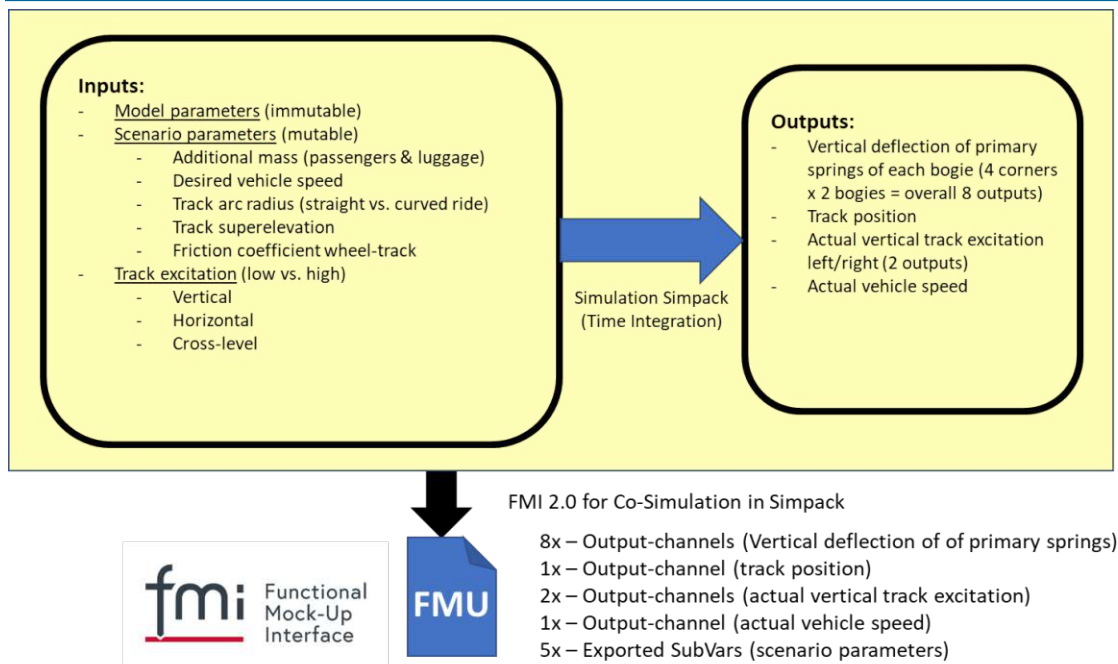


Table 5-1: Scenario parameters for the MBS vehicle.

Scenario Parameter	Parameter Variation						
Track arc radius r [m]	300	500	700	900	1.200	1.500	10.000 (Straight ride)
Track superelevation u [m]	0,15	0,15	0,10	0,10	0,05	0	0
Vehicle speed [m/s]	$v = \sqrt{r \times (0,6 + 9,81 \times (u/1,5))}$						120 / 3,6

Additional mass [kg]	Very crowded (approx. 200 passengers ≈ +15.000 kg)	Little crowded (approx. 100 passengers, ≈ +7.500 kg)	No passenger
Friction coefficient [-]	0,10 (winter: rainy, wet)		0,35 (summer: dry)

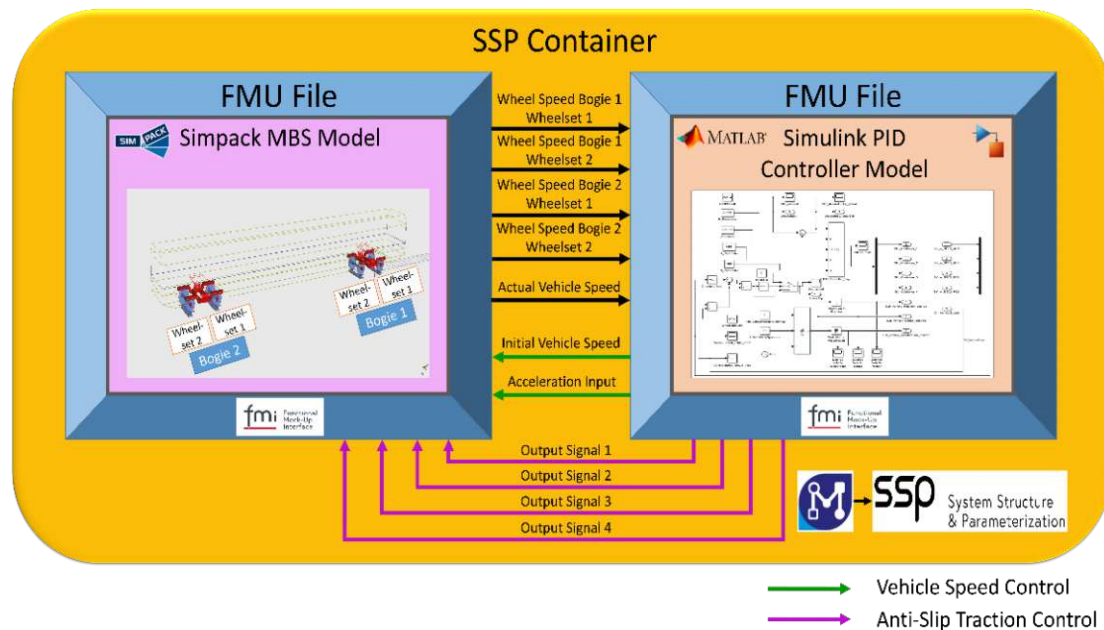
5.1.3 Anti-Slip Traction and Vehicle Speed Control of an MBS Railway Vehicle

This use case is derived from the MBS vehicle model and control engineering. The aim of this use case is to control the anti-slip traction behavior and vehicle speed of the MBS model in the R4F Platform. Wurth [141] succeeded the basics for the implementation of this use case into the platform as a co-simulation model for the Rail4Future project. This model consists of two sub-models, one of which is the Simpack MBS model (see [Subsection 5.1.2](#)) and the other one is a PID-based control model modeled in MATLAB/Simulink. First, these two sub-models are packaged into two FMU files using the Simpack tool for the MBS model, and FMI Kit for Simulink open-source software package [40]. Then, the FMUs of these two sub-models are connected to each other and their co-simulation is tested in the commercial Model.CONNECT co-simulation and integration tool provided from AVL List GmbH. In this tool, the simulation results and inputs can be analyzed in real-time, and CSV files can also be generated from the results. After the simulation testing and result analysis, the entire co-simulation model is packaged into SSP, which contains an SSD system structure description file, both the FMU files and two SSV parameter description files of these FMUs. The SSP file is able to be simulated as a black-box in the platform using some particular open-source software packages such as fmpy, vico [137] and OpenMCX [95] as previously found out in this dissertation project. The fmpy Python package is chosen for this task in order to fulfill the general solution approach regarding to the Rail4Future project. The fmpy package was also used to simulate the MBS vehicle, RLT bridge and ML-based surrogate model in the platform. This co-simulation use case was successfully integrated and then processed in the DT platform by following the proposed centralized co-simulation implementation technique (see Figure 4-5) while utilizing the

FMI/SSP standards and the Python package in this technique (see Kugu et al. [77]). Besides, the automation and management of the integration and processing of the entire model in the platform were succeeded by using the CI/CD pipeline technology [77].

Figure 5-4 shows exactly the FMI- and SSP-based containerization approach, suggested by Kugu et al. [76], based on the model standardization approach (see Figure 4-1), and previously mentioned for the ML-based Reinforcement Learning (RL) parameter tuning methodology by Zhou et al. [148], for the implementation of this use case into the platform. As realized in this figure, input and output data flows occur due to the anti-slip traction behavior and vehicle speed control purposes. For the vehicle speed control, the initial speed in km/h and acceleration in m/s^2 can directly be entered into the Simulink PID-based control model by the users. On the other hand, the anti-slip traction control succeeds during the simulation by time. For this, first, the Proportional (P), Integral (I), Derival (D) control parameters are specified through the use of the rule of thumb, which is a method following proximity-based practices rather than theory. Then, these controller parameters and the desired slip are given into the controller model as inputs, that is important to stabilize the rail vehicle trip in the virtual environment as experienced for the Rail4Future project before. Besides, it is important to choose the right simulation solver and set the proper step size for the stabilization task as previously found out in this work. In [Subsection 5.2.1.3](#), it is notable that this is successfully done by Kugu et al. [77] reducing the simulation step size. For the anti-slip traction control, the wheel speed of each wheelset, and the actual vehicle speed of the MBS vehicle flow into the Simulink control model and then the output signals, resulting from the PID controller of the Simulink model, flow into the Simpack MBS model. These signals directly manipulate all the wheelsets of the vehicle for its stabilized anti-slip traction behavior control. The entire co-simulation occurs in a closed loop to ensure both the vehicle control types. Other than the above mentioned input parameters, the same parameters of the MBS vehicle are given as inputs by the users, which are the wheel-track friction coefficient, additional mass (passenger + luggage, in kg), track arc radius in m and superelevation in m. Otherwise, the N constant value of the PID controller can be entered into the Simulink control model.

Figure 5-4: A structural overview of the anti-slip traction and vehicle speed control co-simulation model as containerized using the FMI and SSP standards. [77]



5.1.4 ML-based Surrogate Model of an MBS Railway Vehicle

This use case provides valuable insights into the model reduction strategy, which helps to significantly decrease the simulation runtime of the MBS vehicle model while consistently and reliably predicting the vehicle-track dynamic behavior of the model due to different track irregularities at the same time. This is performed by Zhou et al. [149], who successfully prepared, trained and tested the model using the ML-based surrogate modelling methodology incl. an ML model with the NARX network technology. After that, they validated the results of the surrogate model, which are relatively consistent (see [Subsection 5.2.1.4](#)).

Figure 5-5: ML-based surrogate modelling of the Simpack MBS railway vehicle model. [149]

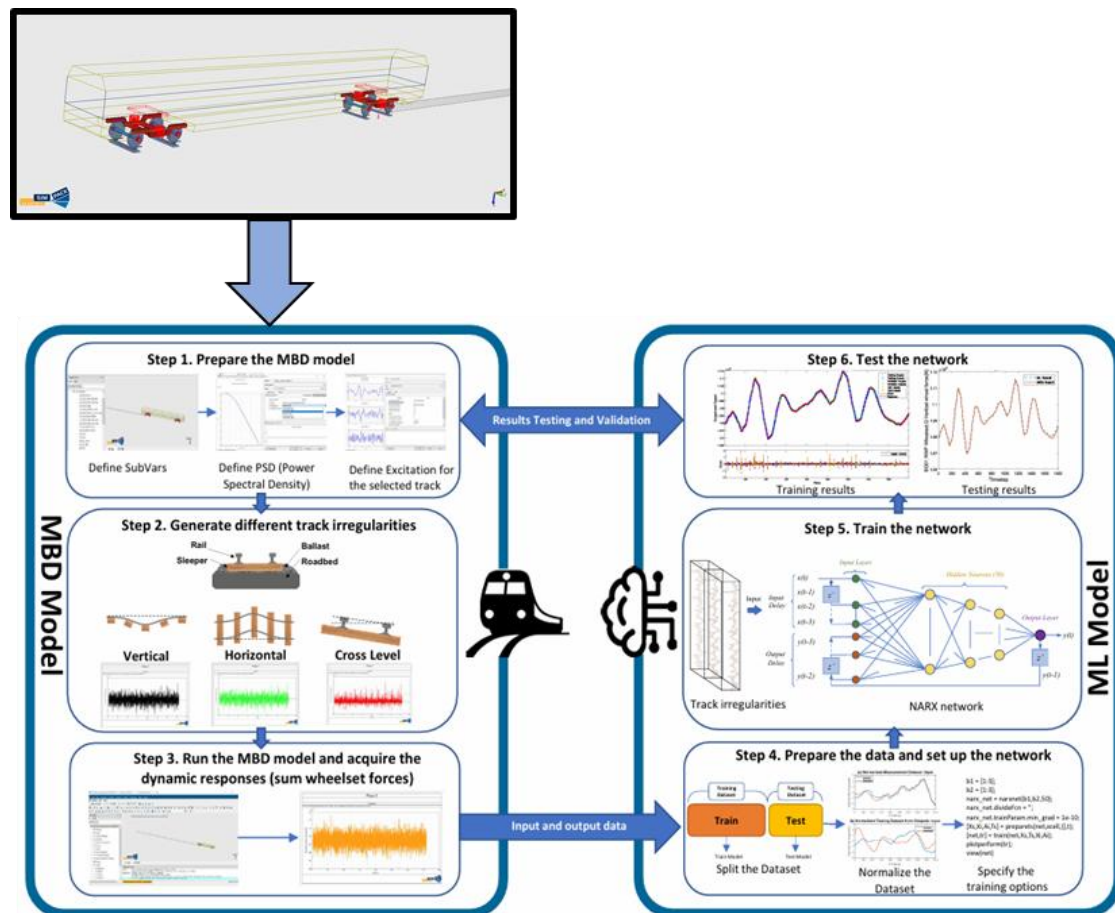
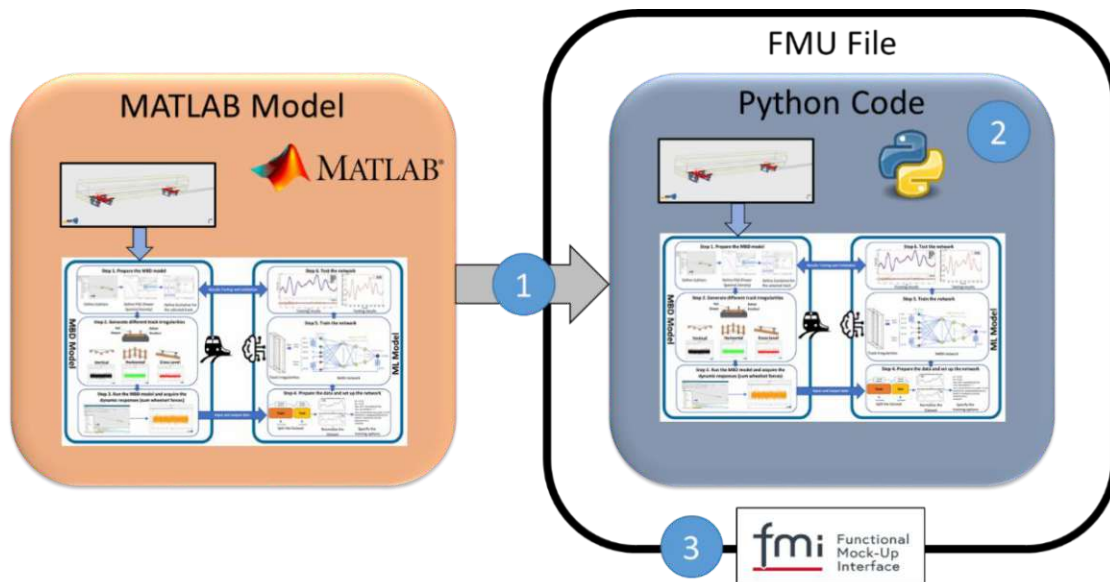


Figure 5-5 shows and describes the steps of the ML-based surrogate modeling approach, proposed by Zhou et al. [149]. These are followed to output the MBS vehicle model as a surrogate model, which is predictively consistent with the MBS model, and serves as a computation time savior for the vehicle simulation (for more information about these steps see Zhou et al. [149]). As they previously achieved, the simulation of the surrogate model over 5 km track takes about 8 seconds, while the simulation time duration of the traditional MBS vehicle model over the same track length is 30 minutes. This means more than 99,50% reduction of the simulation runtime in the R4F Platform.

Figure 5-6: An approach to integrate the ML-based surrogate model into the R4F Platform.



In Figure 5-6, an overview of the model integration approach, used to integrate the ML-based surrogate model into the R4F Platform, is implied. This approach simply consists of three steps. The first step implies the conversion of the surrogate model as a MATLAB code into a Python code, which is succeeded by Zhou et al. [149]. The reason for this conversion is that Python is open-source and relevant to the AIP methodology (see [Subsection 4.2](#)). In the second step, the raw Python code is prepared for the FMU packaging like in the RLT bridge use case. For this, the manual code processing approach, mentioned in Figure 4-2, is simply followed. After the code preparation process, the third step provides the method to package the Python code into FMU, based on the proposed model standardization approach in Figure 4-1, using the *PythonFMU* Python package [55] as previously applied to the RLT bridge Python model.

After obtaining the FMU of this ML-based surrogate vehicle model, the input parameterization of the FMU file is configured by writing a simulation code script in Python, which reads a JSON file as input, and then generates results as validation curves and tables for data analysis after performing the FMI-based simulation of the model. In the JSON file, the users of the R4F Platform simply enter a folder name as input, which indicates a folder containing several CSV

tables. The reason for the folder name entering is that these CSV tables possess high data complexity like the CSV inputs of the RLT bridge use case, mentioned in [Subsection 5.1.1](#), and thus are not directly given into the FMU. These tables include input and output measurement data, which are left- and right-side track irregularities tending in vertical, horizontal and cross directions over a particular track length previously given as inputs and then resulted as outputs in the Simpack tool.

5.2 Results and Discussion

5.2.1 Use Case Results

Besides validating the CI/CD pipeline and system architecture of the test environment by qualitatively comparing the FMI-containerized and raw model simulation outputs through some visual representations, these results are quantitatively evaluated to precisely prove the consistency between the FMI-based and raw model simulation processes in this work. For this, both the root mean square error (RMSE) and mean absolute error (MAE) methods are applied. These methods are given as mathematical formulas and are widely used to indicate the model performance by measuring the deviation between normal and predicted values (see Formula 6.1 and 6.2). On one hand, the RMSE is sensitive to outliers and ideal for normal (Gaussian) distributions. On the other hand, the MAE is less outlier-sensitive than the RMSE, and preferred for Laplacian distributions. [63]

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2} \quad (6.1)$$

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad (6.2)$$

In both formulas, n represents the number of result values per each simulation. i shows the index number of these values. y_i indicates the predicted number, which is replaced by the FMI-based simulation outputs, and \hat{y}_i the true number, representing the raw model simulation results in the case of this work. If RMSE or MAE is zero, it means that the prediction is exactly the same as the normal, namely a perfect match between the FMI-based and raw model simulation results. Also, the closer to zero the error is, the better fit the simulation shows,

occurring in the workflow-based pipeline and system architecture of the test environment. Besides, these RMSEs and MAEs are normalized either to the mean true value or to the interval between the maximum (max) and minimum (min) true values depending on the results, which helps to overcome the scale dependency. According to that, these are called normalized RMSE (nRMSE) and normalized MAE (nMAE). To achieve these, the RMSEs and MAEs are divided by either the mean value or the difference between the max and min values. These normalized values are then multiplied by 100% to express these in percentage. Based on above, the approximation of these nMAE and nRMSE percentage numbers to 0% helps to evaluate the consistency between the FMI-containerized and raw model simulation outputs. [125]

5.2.1.1 RLT Bridge Results

In this subsection, the life times in years, and damage sums in -/year are presented, which are resulted from the RLT calculation of a railway steel bridge occurring in the R4F Platform. Besides, the Python-based raw model and FMI-based SSP (direct FMU) simulation results of both outputs are compared to each other for validation purposes. These results are given as validation points due to the critical points mentioned in [Subsection 5.1.1](#).

Figure 5-7: Life time and damage sum results of the RLT calculation of the dummy bridge delivered from AIT. (Python vs. FMI-based SSP simulation results)

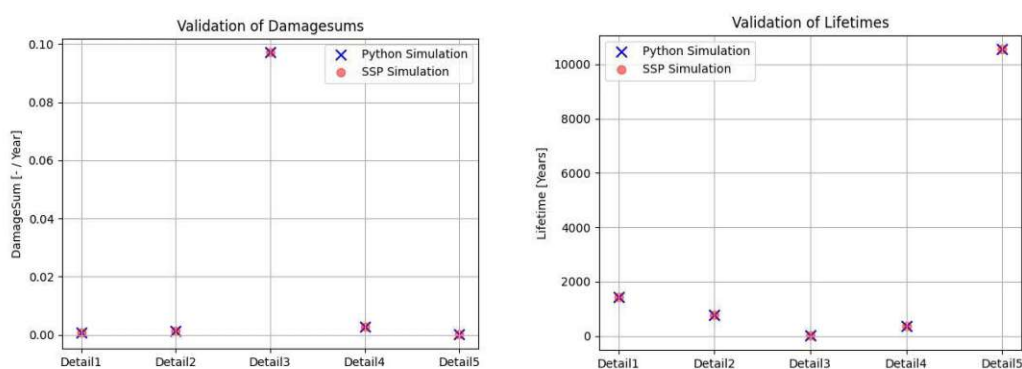


Figure 5-7 indicates the comparison points of the Python-based raw model and FMI-based SSP simulation results coming from the RLT calculation of the dummy railway steel bridge, mentioned in [Subsection 5.1.1](#). As realized in this figure, the FMI-based SSP simulation results shows relatively consistent behavior with the Python outputs, resulted from the direct Python code

execution. To quantitatively evaluate the consistency, the MAE is taken into account because of the Laplacian-kind distribution of these results as discrete points. MAE is exactly zero, which is also 0% nMAE proving the 100% consistency between the FMI-based SSP and Python-based raw model simulation results.

Figure 5-8: Life time and damage sum results of the Eschenau bridge from ÖBB Infrastruktur AG. (Python vs. FMU simulation results)

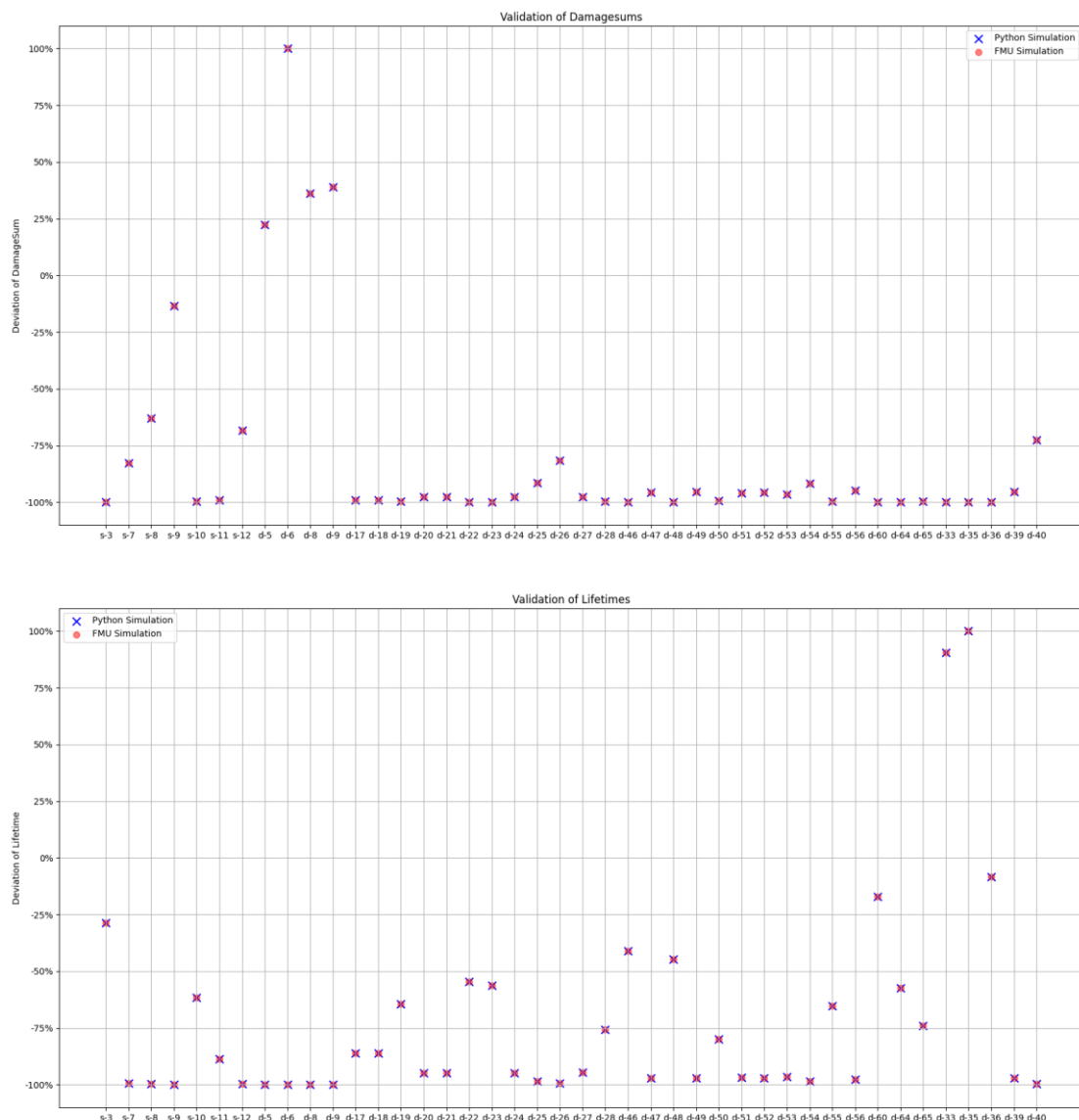


Figure 5-9: Life time and damage sum results of the Mürzbrücke from ÖBB Infrastruktur AG. (Python vs. FMU simulation results)

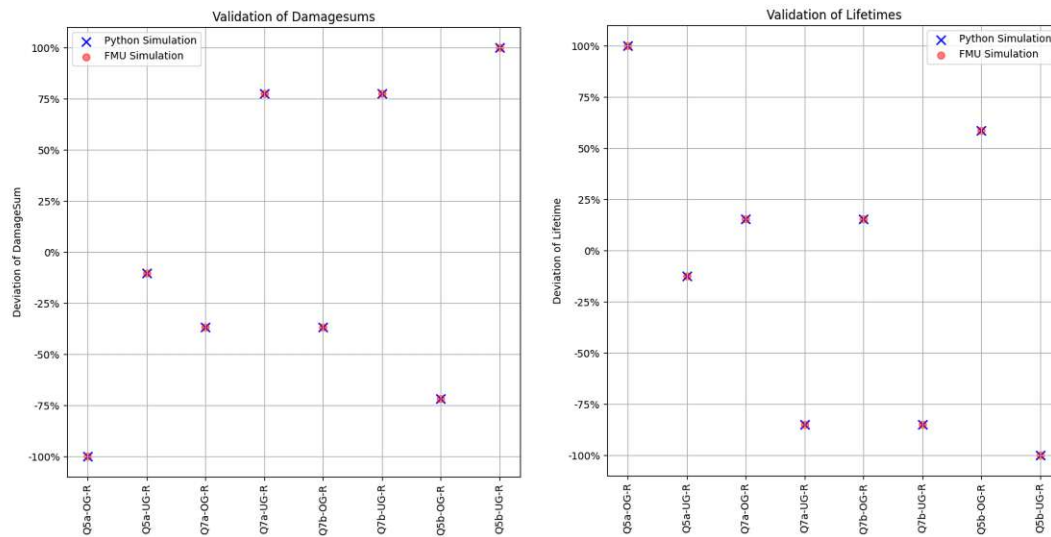
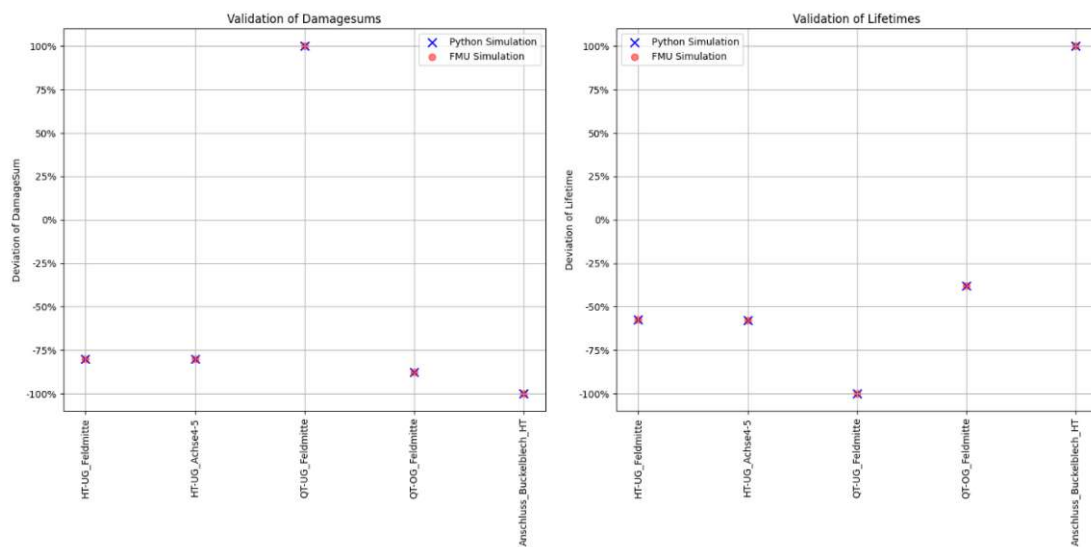


Figure 5-10: Life time and damage sum results of the Schellhamnergasse bridge from Wiener Linien GmbH. (Python vs. FMU simulation results)



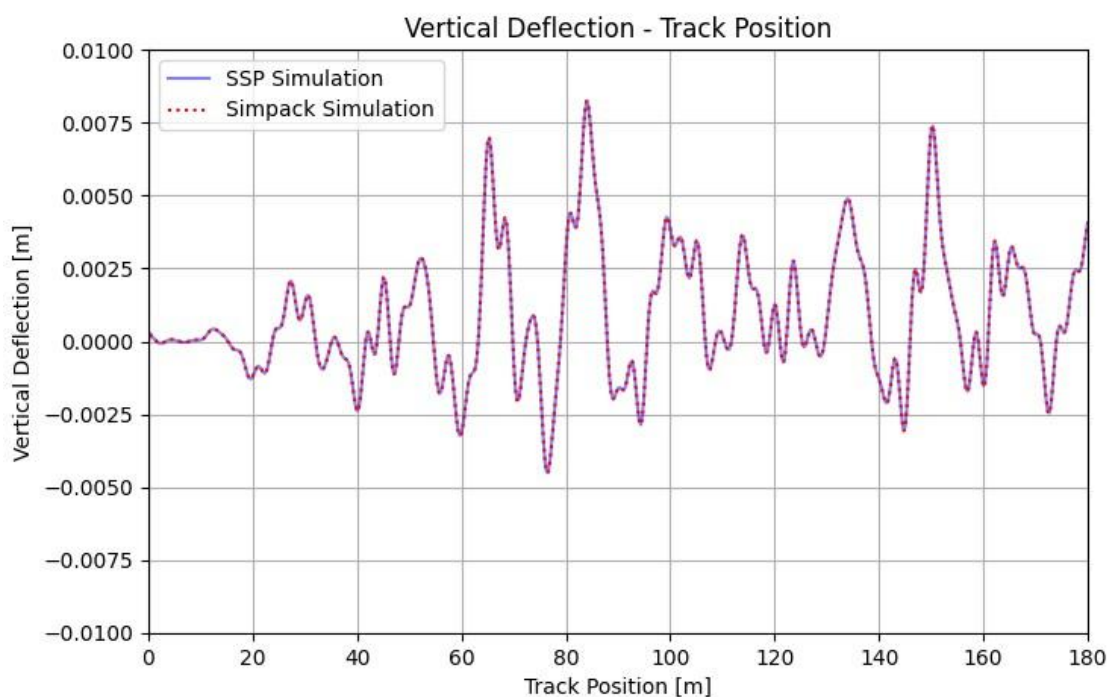
In Figure 5-8, 5-9 and 5-10, all the results of the three real bridge use cases, previously mentioned in [Subsection 5.1.1](#), are presented as validation points.

Due to data protection reasons, it was decided to proceed with the normalization of these result data. These data are normalized to the range of [-100%, +100%] based on the center point (= 0%) to clearly determine the percentage deviation of the results from the center point. As realized in all of these figures, life time and damage sum outputs, resulted from the FMU simulation in the R4F Platform, show relatively high consistency with the results, which come from the direct Python simulation executed in a default simulation tool qualified for Python code executions. The MAE of all these Laplacian-distributed life time and damage sum outputs is exactly zero, which means 0% nMAE and quantitatively proves the high match between the FMU and Python-based raw model simulation results.

5.2.1.2 MBS Vehicle Results

In this subsection, the vertical deflection results of a primary spring located in a bogie belonging to the Simpack MBS vehicle are presented as a result example, where FMI-based SSP (direct FMU) simulation outputs, extracted from the workflow-based pipeline and system architecture of the test environment, are compared with the Simpack-based raw model simulation outputs. These results are then used for further life time calculation of the vehicle in the platform within the Rail4Future project.

Figure 5-11: Vertical deflection results of a primary spring located in a bogie belonging to the MBS vehicle from ViV. (Simpack vs. FMI-based SSP simulation results)



In Figure 5-11, the result comparison is clearly displayed, where the vertical deflection [m] in which track position [m] results due to track irregularities. As realized in this figure, the FMI-based SSP simulation outputs show relatively high consistency with the default simulation output, which means simulation reliability of the platform. The nRMSE of these normally distributed results is 0,27%, namely close to 0%, which quantitatively proves this high consistency. Besides, it is worthy of note that the vertical deflection behaves in an oscillating manner, because track irregularities were also given as inputs in the raw model using the Simpact tool to enhance the realism of the scenarios, mentioned in [Subsection 5.1.2](#), in a virtual environment.

5.2.1.3 Anti-Slip Traction and Vehicle Speed Control Co-Simulation Results

In this subsection, the actual slip, vehicle speed and wheel speed results of the Simpact MBS simulation interacted with a Simulink PID-based control model are presented, where the FMI-based SSP simulation outputs, coming from the

workflow-based pipeline and system architecture of the test environment, are compared with the SIMAT-based raw model simulation outputs.

Figure 5-12: Actual wheel slip, vehicle speed and wheel speed results of the anti-slip co-simulation model. (SIMAT vs. SSP simulation results)

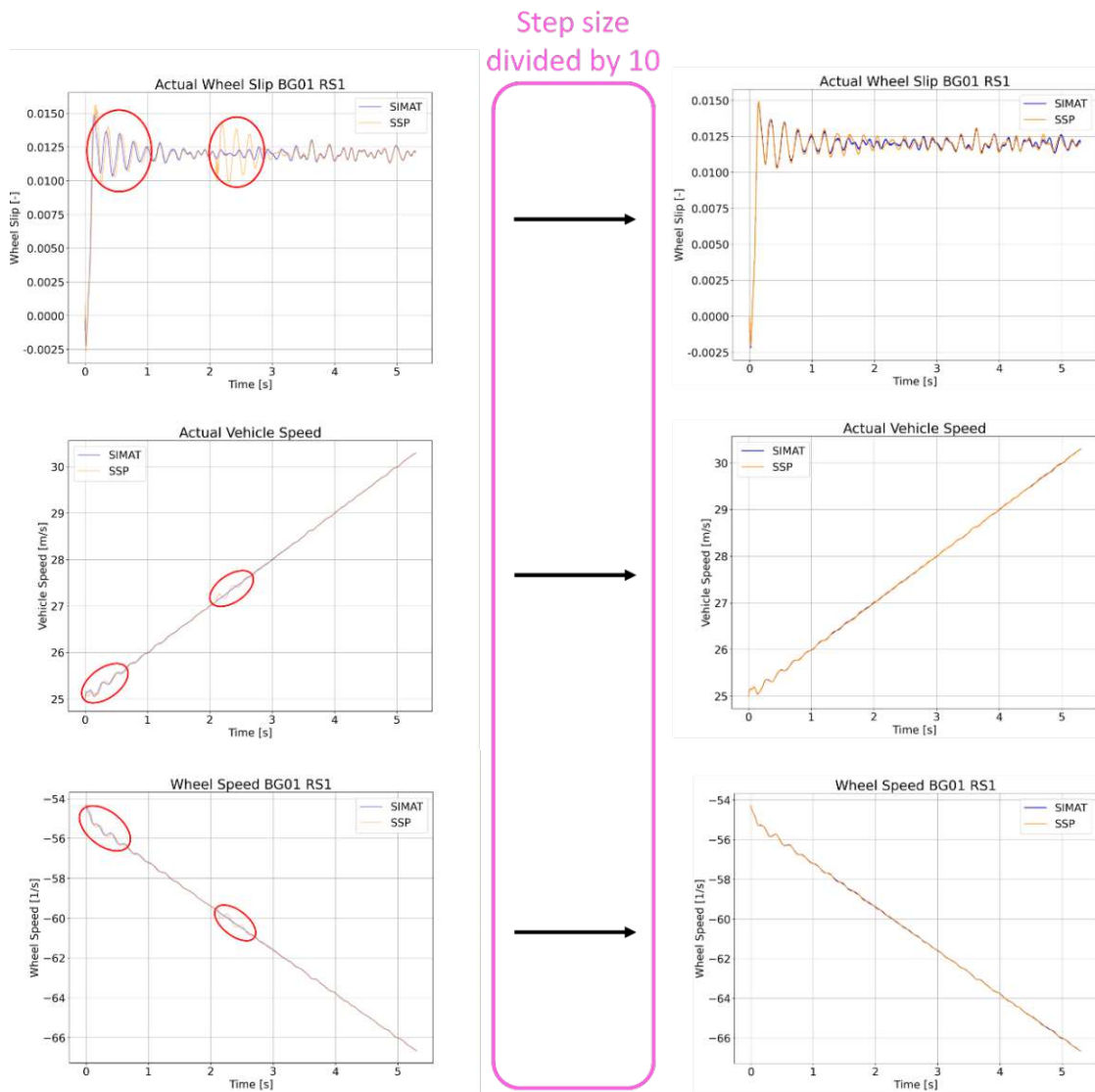


Figure 5-12 indicates the simulation results and their optimization. As realized in this figure, all the three results, coming from the SSP simulation, show relatively consistent behavior with their original simulation results, which originate from the SIMAT simulation. However, unexpectedly occurring oscillations in certain time intervals offend the eye, which indicate a disturbance

of the riding stability of the MBS vehicle in a virtual environment as Wurth [141] found out in his work. To alleviate the disturbance resp. to optimize the simulation results, several solutions were tested and found out useful. In this result optimization, the simulation step size is decreased, which helps to significantly reduce these oscillations as discovered in this work (shown in red circles). In addition to that, Wurth [141] suggested to choose an ideal simulation solver for the optimization task. Therefore, the simulation step size of the FMI-based SSP simulation, executed in the test environment, is decreased by a factor of ten, that leads to more stabilization and also higher simulation runtime (see Kugu et al. [77]). Besides, Wurth [141] chose the Dormand-Prince solver, used to solve ordinary differential equations (ODE), for his work.

To quantitatively evaluate the result consistency and optimization, the nRMSE percentage values of all the Gaussian-distributed results are calculated and compared, shown in Table 5-2. The RMSE values of the wheel slip outputs are normalized to the mean value, which is approx. the desired slip configured by the user. The nRMSEs of vehicle speed outputs are calculated based on the max-min true value difference because of the continuously changing behavior of the vehicle speed value. The nRMSE of wheel speed outputs is not needed because these show the same vehicle acceleration behavior as the vehicle speed results. In this table, first, it is remarkable that all the nRMSE values of these simulation results are near to 0%, proving the high consistency between the FMI-based SSP and raw model SIMAT simulation outputs. Second, the RMSE values of these outputs with smaller simulation step size clearly show lower deviation from 0% than the ones with bigger step size. This quantitatively validates the extraction of less oscillating, namely more optimized results, which are closer to the raw model simulation outputs, and thus proves more stable anti-slip traction and vehicle speed control of the virtual vehicle coming from the step size reduction strategy.

Table 5-2: nRMSE percentage values between the raw model (SIMAT) and FMI-based SSP simulation results.

Simulation Results	nRMSE SIMAT-SSP [%]	
		Step size divided by 10
Wheel Slip	4,788	1,522

Vehicle Speed	0,432	0,127
---------------	-------	-------

5.2.1.4 ML-based Surrogate Model Results

This subsection gives comprehensive insights into the validation of the MBS vehicle model with its ML-based surrogate modelling version. The simulation results of the surrogate model, converted from MATLAB into a Python code by Zhou et al. [149], is simply compared to the simulation results coming from the default Simpack tool to prove the reliability of the ML-based model.

Figure 5-13: Validation results of the ML-based surrogate MBS vehicle model as normalized (predicted vs. real).

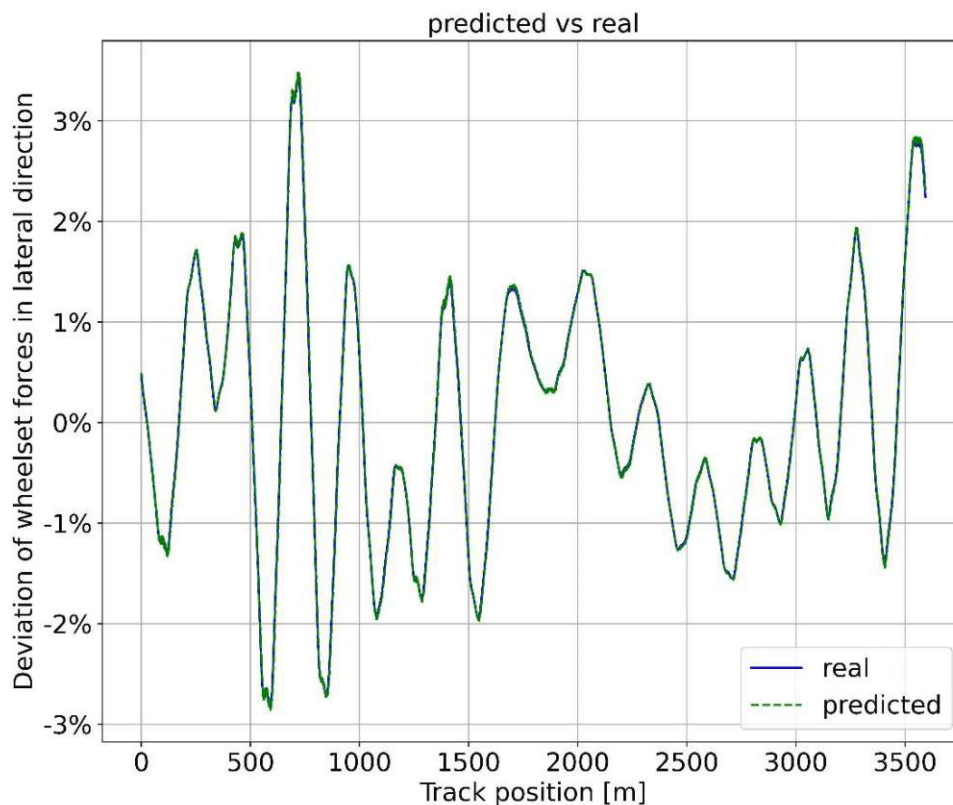


Figure 5-13 indicates the comparison results of actual wheelset forces in lateral direction between the ML-based surrogate model simulation (predicted), described in [Subsection 5.1.4](#), and measurements (real) over several thousand meters of track. Due to data protection reasons, it was decided to proceed with the normalization of the result data. These data are normalized to the range of [-100%, +100%] based on the mean value (= 0%) to clearly determine the

percentage deviation of the results from the mean value. As simply realized in this figure, these results are relatively consistent to each other in spite of little deviations unexpectedly occurred in max and min points. Zhou et al. [149] also proved the high result consistency by calculating the normalized RMSE (RMSE divided by the difference between the max and min values, namely by the interval of the result data [125]) of these results, which is 3,52% and below 10% meaning tolerable by industry partners. The same figure is also achieved from the FMU simulation of the ML surrogate model, which occurs in the workflow-based pipeline and system architecture of the test environment. This clearly shows the accurate behavior of the ML-based model, as well as the simulation reliability of the platform, which encourages the integration and processing of the ML surrogate simulation asset in the platform for the Rail4Future project.

5.2.2 Discussion

5.2.2.1 Benefits of the Proposed Methodology and System Architecture

This subsection clearly states significant advantages of the proposed methodology and system architecture (see [Section 4](#)) for the AIP, its automation & management and co-simulation implementation tasks.

- **AIP methodology:** First, it is worthy to point out that this methodology is designed and developed at open-source level, which means utilizing key technologies such as open-source interface standards, OOP, VM, other software tools, libraries and packages. Besides, the methodology assisted to control the quality, version, portability and release of different railway simulation assets, to ensure their simulation tests incl. simple user input parameterization and adequate output generation in a railway DT framework such as the R4F Platform. This helped to adapt different simulation assets to such a framework by following the methodology.
- **AIP automation & management approach:** This approach, which is based on a workflow and system architecture, aimed to improve the AIP task by automating and easing its management at open-source level in terms of time-, money-efficiency, quality & version control, team collaboration, workflow orientation and practicability. Kugu et al. [77] mentioned these beneficial aspects supporting the proposed automation & management methodology.
- **Co-simulation implementation methodology:** First, this methodology, based on the proposed AIP methodology and system architecture, helped to add more functionality to simulations of railway subsystems such as the MBS railway vehicle, the speed and anti-slip traction of which

are successfully controlled in a railway DT environment (see [Subsection 5.1.3](#)). Then, portability, quality, version, delivery and simulation tests incl. easy user input parameterization and suitable output generation are managed by following the proposed approach of the methodology for centralized co-simulation implementation into a railway DT framework such as the R4F Platform. Lastly, the distributed co-simulation approach of the methodology was proposed as an alternative solution to the centralized approach, which significantly contributed to resource consumption & software license management, open-source software tendency, result optimization, IP protection of simulation models by splitting simulations into different resources as discovered in this work (for more information see [Subsection 5.2.2.6](#) of this section).

5.2.2.2 Software License Management

As discovered in this dissertation project, the management of software licenses cannot be denied even if the open-source level is strived for the design and development of the entire methodology. The reason for this is that some of the railway use cases is still dependent on software licenses, which authorize the users to execute the simulations of the use cases in a railway DT platform such as the R4F Platform. For example, the Simpack license was necessary together with the Simpack solver to run the simulations of the MBS vehicle and its extended anti-slip co-simulation use cases in the platform. Besides, a Virtual Private Network connection was needed to be established in the platform to reliably, securely connect it to the corresponding license server, belonging to the license owner. Moreover, the MATLAB license was used to be able to build the PID-based control model, and package this into FMU in conjunction with the use of the FMI Kit for Simulink tool for the co-simulation use case (see Wurth [141]). Furthermore, the Model.CONNECT software license was utilized for the co-simulation use case, which helped to link FMUs to each other through connections, then test the FMI-based co-simulation and finally create an SSP file from this in the software platform.

5.2.2.3 Software Installation and Configuration

As realized in this work, several key technologies incl. interface standards, software tools, applications, libraries and packages are involved to build and develop the entire proposed methodology and system architecture. Depending on this, specific software installations and configurations needed to be done to effectively design and develop the workflow-based pipeline and system architecture of the test environment, representing a simplistic and compatible

version of the R4F Platform, also the entire methodology for the research of this work. This definitely requires relatively comprehensive know-how and experience about all the software. Kugu et al. [77] also mentioned the discussion matter related to pipeline installation and configuration for the AIP automation and management as a concrete example.

5.2.2.4 Platform and Software Compatibility

The platform compatibility is one of the key aspects to take into account for applying simulation assets of the railway use cases to the test environment in this work. For example, the OS of the binary of an FMU is compared to the OS of the test environment to determine the FMU's ability for its execution in the environment. Besides, software compatibility plays a huge role in selecting specific software tools, applications, libraries and packages for the design and development of the test environment, which supplies tool support for the AIP, automation & management and co-simulation implementation tasks. For example, an FMI-compatible and Python-related tool had to be selected to package a Python model into the FMU such as in the RLT bridge and ML-based surrogate modelling use case examples, where the *PythonFMU* library [55] was found in the FMI Tools webpage [133], chosen and then utilized for the FMU containerization task.

5.2.2.5 Use Case Specific Limitations

Speaking about the use cases, some particular obstacles, based on these use cases, are discovered while integrating and processing their simulation assets in the test environment. For example, the results of the anti-slip traction and vehicle speed control co-simulation, shown in [Subsection 5.2.1.3](#), show unexpectedly occurring oscillating behavior meaning instabilities in the anti-slip traction control. Besides, the co-simulation of the MBS vehicle with the control model with the refined simulation step size, which is important to achieve more stability, is relatively time-consuming even over a couple of hundred meters of track. This means a trade-off between simulation runtime and result quality.

5.2.2.6 Centralized vs. Distributed Co-Simulation

In order to finish the Rail4Future project by its end time (30. September 2024), the SSP container-based centralized co-simulation was succeeded, because the trend lied on making different railway simulation models portable as a file for this project. As previously shown and described in [Subsection 4.2](#), the containerization of these models is succeeded using the FMI and SSP standards, which provide more description, file portability, tool independency,

model connectivity and platform compatibility. However, it is discovered later that these models, containerized into the FMU, might be co-simulated in a distributed fashion using the OSP software application, for which a distributed co-simulation methodology is proposed as previously mentioned in [Subsection 4.4](#). Besides, it was found out that the distributed co-simulation technique helps to reduce the resource consumption in general through the use of the IP/port-based networked connection technology, which the OSP software utilizes. This was proved by measuring and then quantitatively evaluating the CPU and RAM of subprocesses, which are needed for the co-simulation process taking several minutes, running in computer machines. The measurement was simply performed by using an internal monitoring tool of the machines, from where measured values are directly extracted per each minute. The quantitative evaluation is based on the mean value, which is obtained by dividing the sum of these values by the simulation duration rounded to the minute when the last measured value is obtained. This then helps to receive average CPU and RAM results, which are shown in Figure 5-15 and Table 5-3 in this work.

Figure 5-14 indicates actual wheel slip, actual vehicle speed and wheel speed results, where the OSP-based distributed co-simulation results are compared to the SSP and SIMAT co-simulation outputs with the same result optimization shown in Figure 5-12. As clearly realized in Figure 5-14, the OSP-based distributed co-simulation process shows relatively stable behavior from the beginning compared to the SIMAT and SSP-container based processes, which were previously succeeded for the Rail4Future project, at the same simulation step size in general.

Figure 5-14: Comparison of simulation results of the anti-slip co-simulation model to the outputs resulting from the OSP-based distributed co-simulation process. (SIMAT vs. OSP vs. SSP)

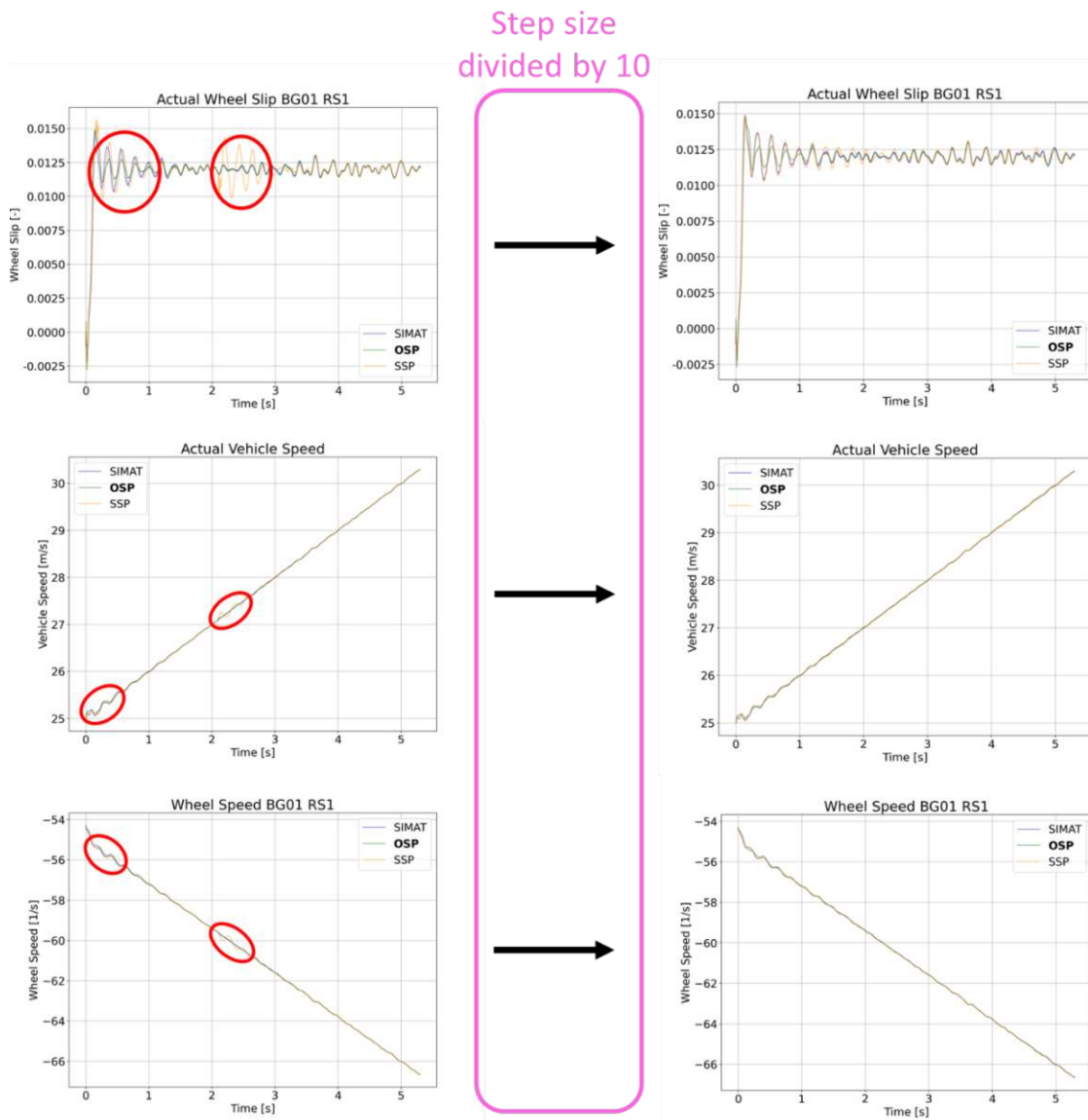


Figure 5-15 exposes comparative insightful results, visualized using the Microsoft Excel tool, and these results come from the computational resource consumption, caused by the SSP centralized and OSP distributed co-simulation of the anti-slip traction and vehicle speed control co-simulation model after the

previously mentioned result optimization. First, it is important to notice that each VM is assigned to one role (Asset Integrator or Asset Provider, mentioned in [Subsection 1.5](#)) due to the distributed co-simulation technique (see [Subsection 4.4](#)), and set to the same CPU and RAM for the reliability and transparency of the resource consumption test. On the upper part of this figure, CPU usage of all sub-processes, occurring during the simulation process in one or three VMs, is clearly shown in 2D pie charts. The CPU usage indicates the load quantity, utilized by processor cores to start different applications on a computer, thus it can considerably affect the computer's performance. As realized in this figure, the CPU load is clearly lightened by distributing the *FMI2Process* sub-process, coming from the FMI-based simulation of the MBS vehicle, from the Asset Integrator's VM (VM 1) to other two VMs of the Asset Provider(s) (VM 2 & VM 3) through the use of the *cosim* and *proxyfmu* tools (*cosim* & *proxyfmu* sub-processes), mentioned in [Appendix Subsection A.2.4](#), in the distributed co-simulation. In the meanwhile, the essential sub-processes *python*, which comes out through the use of the Python tool for the asset application (see [Subsection 4.2](#)), and *FMI2Process* need to occur centrally in VM 1 for the SSP container-based centralized co-simulation. On the lower part of this figure, the RAM usage of all the sub-processes, belonging to the two co-simulation types, in the VMs is visually indicated in 2D bar charts. The RAM usage refers to the consumption of the available electronic computer memory, which can significantly affect the performance of the computer. First, the extreme RAM usage of the *python* sub-processor for the SSP-container based co-simulation is noticeable compared to the other sub-processors in both co-simulation types. Therefore, the distributed co-simulation causes definitely lower RAM usage than the SSP co-simulation in sum as clearly realized from the unremarkable effect of the *cosim* and *proxyfmu* sub-processes on the general RAM usage in all the three VMs.

Figure 5-15: CPU and RAM average resource consumption results. (SSP Centralized vs. OSP Distributed Co-Simulation)

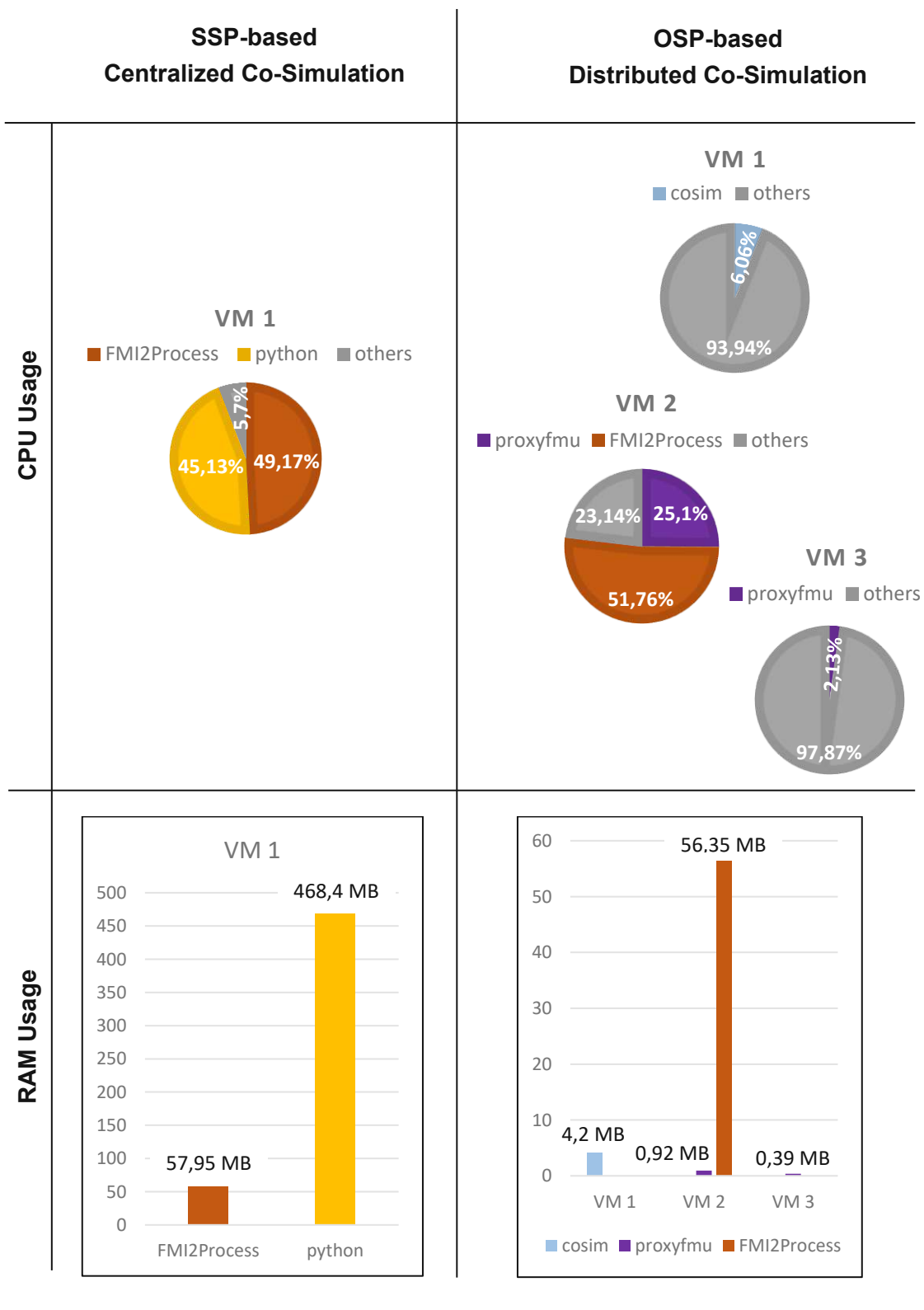





Table 5-3 shows the total CPU, RAM and storage space average results of the two co-simulation types in all the three VMs as numbers. As directly understood from the CPU usage reduction by approx. 8,6%, the distribution of the CPU usage due to the distributed co-simulation contributes to the resource efficiency of the VM 1. Besides, the RAM usage in the distributed co-simulation is extremely decreased by approx. 88,25% compared to the SSP simulation, which makes the distributed co-simulation more preferable than the other one from the RAM usage perspective. Lastly, the storage space, needed for the implementation of these co-simulation types, is also noticed. First of all, it is important to point out that remarkably less storage space in the Asset Integrator's VM is needed for the distributed co-simulation compared to the SSP simulation case. The reason for this is that all the software tools, licenses, libraries and packages, which are needed to instantiate the FMU files, are directly installed on VM 2 and VM 3 instead of putting all these into VM 1. Besides, the difference between the total storage spaces of the co-simulation types (approx. 8,4 %) mainly lies on the multiplication of the FMU files of the co-simulation use case resulted from the transfer of the FMUs to VM 2 and VM 3 in the distributed co-simulation case.

Table 5-3: Resource consumption results (average).

Computer Machines		SSP Centralized Co-Simulation	OSP Distributed Co-Simulation
VM 1	CPU [-]	1,86	0,12
	RAM [MB]	526,35	4,2
	Storage Space [MB]	> 2157,7	≈ 146
VM 2	CPU [-]	X	1,54
	RAM [MB]	X	57,27
	Storage Space [MB]	X	> 2150,7
VM 3	CPU [-]	X	0,04
	RAM [MB]	X	0,39
	Storage Space [MB]	X	≈ 42,09

Total CPU	1,86		-8,6%	1,70
Total RAM	526,35 MB		-88,25%	61,86 MB
Total Storage Space	> 2157,7 MB		+8,4%	> 2338,79 MB

Based on the above, the aspects such as optimal simulation results from Figure 5-14, computation resource efficiency, open-source software level and software license management make the distributed co-simulation methodology more profitable than the SSP-based centralized co-simulation technique. As previously found out, the OSP-based co-simulation technique could help to partially enhance the IP-protection of the simulation models, because the models are not distributed from the master machine into third party slave machines, which are other than the ones providing the models, however these models need to be shared as FMUs between the master machine and the corresponding slave machines in case of distributed co-simulation. On the other hand, the SSP centralized co-simulation technique is easier and less complex to implement and manage than the OSP distributed co-simulation methodology regarding to the file portability as one SSP container, containing all the FMUs and descriptions, and lower number of computer machines, needed to be set up for the implementation and management task. The distributed co-simulation technique is successfully implemented into a virtual platform, which might be the R4F Platform, as an alternative solution to the centralized co-simulation methodology, based on the FMI/SSP. To apply the distributed co-simulation process into the platform using the OSP implementation approach (see [Subsection 4.4](#)), the network connection between the master and slave machines, as well as the software license management in the corresponding slave machines, where the licenses are necessary to instantiate an FMU of a simulation model modeled in a commercial tool, are to be definitively considered.

Unfortunately, the DCP standard could not be applied to the platform because of limited support issues of the standard, which is supported by five tools [132]. However, the DCP-based co-simulation would be beneficial to enhance the IP protection of these models, which the Asset Providers might keep only in their machine far from the Asset Integrator.

6 Conclusion

This doctor's thesis aimed to fill the gap between the railway simulation assets and a railway DT platform such as the R4F Platform. First, key technologies, which are seen as a potential to integrate and process these assets in the platform at open-source level, such as open-source interface standards, semantic web, OSP software, OOP, CI/CD pipeline, VM, software containerization, CAT and VCS technologies were chosen (see [Section 2.2](#)). These were identified according to the state-of-the-art investigation of this work through a comprehensive literature research (see [Section 3](#)). Then, the AIP task was defined according to the problem statement (see [Subsection 1.2](#)), main objective (see [Subsection 1.3](#)) and state-of-the-art investigation of this work as shown in [Subsection 4.1](#). After that, the entire methodology and system architecture incl. approaches and methods, which are proposed to fulfill the AIP task, to automate and manage the AIP, and finally to implement co-simulations into the platform in a centralized and distributed fashion, were designed and developed (see [Section 4](#)). Furthermore, the use cases, successfully applied to the platform, were shown, described with tables and submodel pictures (e.g., 3D image of the RLT bridge), and their simulation results were presented by providing 2D visual curve & point diagrams, clearly indicating the qualitative and quantitative validation between the raw modelling and pipeline & system architecture of the test environment, in this work (see [Section 5](#)). In addition to that, different matters, encountered during the AIP, its automation & management and co-simulation implementation into the DT, such as benefits of the proposed methodology & system architecture, software license management, software installation & configuration, platform & software compatibility, use case specific limitations and distributed vs. centralized co-simulation were discussed in this work. The distributed co-simulation implementation, succeeded to fulfill the co-simulation implementation task as an alternative solution to the centralized co-simulation, was discussed through its comparison with the centralized approach, which give comprehensive insights into advantages, disadvantages, challenges and limitations of the alternative implementation method (see [Subsection 4.4](#) and [Subsection 5.2.2.6](#)).

Hereby, the RQs of this dissertation work are answered as it follows:

RQ1. What methodologies and key technologies can be applied to realize the asset integration and processing task at open-source level?

A: First, a comprehensive definition of the AIP task should be provided to describe what exactly must be done in terms of integration and processing of simulation assets in a railway DT framework under consideration of some criteria such as description transparency, platform compatibility, tool-independency, version and quality control of these assets. Then, an AIP methodology should be concretized by utilizing key technologies such as the FMI/SSP interface standards, OOP, VM technologies, other software tools, libraries and packages (e.g., *fmpy* Python package [49]) to realize the AIP task in a railway DT platform at open-source level. After that, a centralized co-simulation implementation technique, which is the extended version of the model simulation approach belonging to the AIP methodology, should be utilized to integrate and process railway co-simulation assets centrally in the platform at open-source level by using the FMI/SSP standards, OOP, VM technologies and other software.

RQ2. How can the asset integration and processing task be automated and managed at open-source level? What challenges and limits are expected to be encountered while striving for full automation?

A: As previously found out in this work, the DevOps approach and CI/CD process can be perceived to automate the AIP task by using open-source DevOps- and CI/CD-related software technologies, which are shown in the proposal automation and management approach (see Figure 4-4), in the test environment, representing a simplistic and compatible version of a railway DT framework such as the R4F Platform and therefore used for the research in this work. For this, a proposal system architecture can be utilized to automate and easily manage the AIP task by using a workflow-based CI/CD pipeline in the test environment. The pipeline can be designed, developed and interacted with other potential software technologies such as VCS and CAT, VM, software containerization, other software tools, libraries and packages (e.g., mail transfer agent for notifications) in the platform. To confirm the validity and reliability of the pipeline and system architecture, simulation results, coming from the CI/CD pipeline execution, should be presented, compared with default outputs, and finally the comparison can be quantitatively evaluated by utilizing the MAE and RMSE methods (see [Subsection 5.2.1](#)). There are challenges and limitations regarding to code processing and approval for the asset delivery, which need

to be overcome to reach the full automation level in a railway DT platform such as the R4F Platform. First, it is relatively elaborate to organize the auto-replacement of different components in a Python code using the AI technology, which would help to prepare a Python code for its FMU packaging instead of manually generating the code, shown in Figure 4-2. Of course, the gap between the code and auto code generation would be filled, if an AI-supported code generation tool was extensively tested and finally integrated into the platform. Besides, the approval process for the asset release to other stakeholders might be automated, which was previously succeeded by Reiterer et al. [112]. However, it would be best practice to keep a person in charge at the beginning, who is the Asset Integrator of the platform and takes care of result analysis, security checks and continuous improvement of railway simulation assets by receiving useful feedback from their colleagues. In addition, the R4F Ontology, based on the ontology-based semantic technology, can be utilized to track and comprehend railway simulation assets by searching and finding key information about these (e.g., FMI version number, author name), which helps to ease the management of the AIP task. Besides, a distributed co-simulation approach can be utilized as an alternative solution to the centralized approach, which helps to manage the resource consumption and software licensing by distributing simulation processes from the test environment into different machines/workstations, namely other stakeholders at open-source level.

RQ3. How can co-simulations be distributed into different machines without sharing the simulation models due to intellectual property (IP) protection of the models?

A: A distributed co-simulation approach can be utilized as an alternative solution to the centralized approach to protect the IP of simulation models by distributing simulation processes from the Asset Integrator's test environment, representing a simplistic and compatible version of a railway DT framework, into different machines/workstations, namely other stakeholders at open-source level, which also helps to optimize simulation results, manage the resource consumption and software licensing as remarked in this work. For the realization of the distributed co-simulation process, the OSP and VM software technologies can be utilized in conjunction with the FMI standard. After the distributed co-simulation of the anti-slip traction and vehicle speed control use case was succeeded in the test environment, it was found out that the FMUs of the simulation models are shared with the external machines while splitting their

simulation processes into these machines. This means that the corresponding Asset Provider gets the FMU, packaged from their adapted raw model, indeed, but other Asset Providers do not receive the FMU. On that basis, it is right to say that the IP of the models is partially protected in the distributed co-simulation case. On the other hand, the full IP protection might be achieved by utilizing the DCP standard. However, the distributed co-simulation with the DCP could not be achieved, because the DCP standard has limited tool support issues as previously mentioned.

7 Limitations and Future Work

This dissertation project may be enhanced through networking, data management, full automation, AI and information security in future. Besides, the co-simulation study may be extended for future research.

There will be several platforms, which possess the prototype of the R4F Platform and belong to different stakeholders. These might be connected to each other to ensure the data and model exchange between the stakeholders during conditional monitoring, predictive maintenance and visualization of the railway infrastructure system. For this, the importance of an efficient, reliable and secure network design between these virtual platforms cannot be denied.

Depending on the networking aspect, the data management between these stakeholders could be enhanced through the use of data spaces, which potentially handle the data exchange under consideration of data protection, data governance and data compliance rules.

The full automation of the entire AIP task is a huge trend to further reduce the lifecycle costs of the railway infrastructure system, and improve the time and resource efficiency of the DevOps practices performed in the DT platform. As a concrete example, Reiterer et al. [112] successfully automated the building, testing and deployment of the railway assets in the R4F Platform by succeeding the pipeline auto-generation and -configuration using the graph database technology. At the same time, they managed the auto-checking process helping to validate all the interfaces and results regarding to the asset applications, which are designed, developed and finally applied to the platform for the Rail4Future project. Regarding to the full automation, the IT infrastructure of the R4F Platform might be automatically built, configured, versioned and dependent on this, its resources might be easily managed in future using an open-source infrastructure-as-code software tool (e.g., Terraform).

The AI might improve the AIP by providing auto chat suggestions, which would assist the Asset Integrator to efficiently build, test and deliver further asset applications to other stakeholders. Besides, this AI technology could be extended to other use cases than the ML-based surrogate modelling methodology, succeeded by Zhou et al. [149]. For example, it would be insightful to implement the surrogate MBS vehicle model into the anti-slip

traction behavior control and vehicle speed co-simulation model. By this, the simulation runtime of the co-simulation model would be significantly reduced, which would make it adaptive for large-scale simulations in the platform. Furthermore, the manual code preparation methodology, mentioned in Figure 4-2, could be further automated using a AI-supported code generation tool, which would definitely contribute to the full-automation of the asset integration and processing task as well.

The information security in the platform might be enhanced through the use of some particular AI-based technologies. For example, some parts of the pipeline console outputs of the asset integration and processing, which other stakeholders should not see due for IP protection and security reasons (e.g., user credentials, simulation solver name), could be automatically filtered out or censored using an AI tool.

In future, the distributed co-simulation process, mentioned in [Subsection 4.4](#), might be performed using the DCP standard in the R4F Platform, which would fully enhance the IP protection of the simulation models, packaged into FMUs and applied to the co-simulation process. Therefore, the model exchange between the Asset Integrator and Asset Provider for the distributed co-simulation doesn't need to occur. Besides, adaptations such as the C++ implementation of these models into the DCPLib library [27], which is open-source, written in C++ and supported by the DCP standard, might help to utilize the DCP standard for the distributed co-simulation process in the platform at open-source level. This might be interesting and also extremely challenging. As a reference for an exemplary DCP implementation, Reiterer et al. [111]'s work might be taken as well, where they implemented the DCP master by using Python, and built an FMI-to-DCP wrapper to run FMUs as DCP slaves. Besides, the single FMUs of other use cases are planned to be executed in a distributed fashion without any model sharing in the platform. In addition to that, the OpenModelica Simulator (OMSimulator) [94] open-source software tool might be used to create an SSP system, test the co-simulation of connected FMUs in the system, generate simulation results and finally export an SSP file from the system to stay at the open-source level as later discovered for this work. Regarding to the comparative result analysis between the distributed and centralized co-simulation, mentioned in [Subsection 5.2.2.6](#), different input parameterizations might be taken into account to gain more comprehensive insights into the result consistency and simulation reliability of the distributed

co-simulation process due to various scenarios (e.g., straight vs. curved track, accelerating vs. deaccelerating vehicle). Besides, the resource consumption study could be extended to other resources such as electricity consumption or network usage other than CPU, RAM and storage space. Additionally, the P, I and D controller parameters of the PID-based control model, belonging to the anti-slip traction and vehicle speed control use case, could be specified using the heuristic Ziegler-Nichols method for the centralized and distributed co-simulation. This would help to analyze the co-simulation results from a broader perspective and maybe it would lead to more optimal results regarding to stability and consistency.

8 References

- [1] 3D Viewer App provided by Microsoft Cooperation. URL: <https://apps.microsoft.com/detail/9NBLGGH42THS?hl=en-us&gl=AT&ocid=pdpshare> (visited on 2025-08-15).
- [2] Abrazeh, S., Mohseni, S. R., Zeitouni, M. J., Parvaresh, A., Fathollahi, A., Gheisarnejad, M., & Khooban, M. H. (2022). Virtual hardware-in-the-loop FMU co-simulation based digital twins for heating, ventilation, and air-conditioning (HVAC) systems. *IEEE transactions on emerging topics in computational intelligence*, 7(1), 65-75. URL: <https://doi.org/10.1109/TETCI.2022.3168507>
- [3] Adeagbo, M. O., Wang, S. M., & Ni, Y. Q. (2024). Revamping structural health monitoring of advanced rail transit systems: A paradigmatic shift from digital shadows to digital twins. *Advanced Engineering Informatics*, 61, 102450. URL: <https://doi.org/10.1016/j.aei.2024.102450>
- [4] Ahmad, S., Spiryagin, M., Wu, Q., Bernal, E., Sun, Y., Cole, C., & Makin, B. (2024). Development of a Digital Twin for prediction of rail surface damage in heavy haul railway operations. *Vehicle System Dynamics*, 62(1), 41-66. URL: <https://doi.org/10.1080/00423114.2023.2237620>
- [5] Ahmadi, M., Kaleybar, H. J., Brenna, M., Castelli-Dezza, F., & Carmeli, M. S. (2021, February). Adapting digital twin technology in electric railway power systems. In *2021 12th Power Electronics, Drive Systems, and Technologies Conference (PEDSTC)* (pp. 1-6). IEEE. URL: <https://doi.org/10.1109/PEDSTC52094.2021.9405876>
- [6] Aissat, S., Beaulieu, J., Bordeleau, F., Motamedi, A., Poirier, É. (2024). A DevOps Approach for the Systematic Development and Evolution of Built Assets Digital Twins. *Proceedings of the 41st International Conference of CIB W78, Marrakech, Morocco, 2-3 October, ISSN: 2706-6568.* (ISSN: 2706-6568). URL: <https://itc.scix.net/paper/w78-2024-114> (visited on 2025-08-07).
- [7] Anyene, G., Nepomuceno, A., Kim, I., & Schultz, C. (2023, May). Digital twin and agent-based simulation: co-simulation to support intelligent navigation of healthcare mobile robot. In *2023 Annual Modeling and Simulation Conference (ANNSIM)* (pp. 208-219). IEEE Computer Society. URL: <https://www.computer.org/csdl/proceedings-article/annsim/2023/10155388/1OidulEu9AQ> (visited on 2025-08-08).

- [8] Anyene, G., Schultz, C., Nepomuceno, A., & Kim, I. (2024). Digital-twin co-simulation framework to support informed decision in healthcare planning and management. *Simulation*, 101(3), 361-375. URL: <https://doi.org/10.1177/00375497241283047>
- [9] Armijo, A., & Zamora-Sánchez, D. (2024). Integration of railway bridge structural health monitoring into the internet of things with a digital twin: a case study. *Sensors*, 24(7), 2115. URL: <https://doi.org/10.3390/s24072115>
- [10] Barbie, A. (2025). Agile Continuous Integration Testing of Embedded Software Systems with Digital Twin Prototypes (Doctoral dissertation). URL: <https://doi.org/10.21941/kcss/2025/2>
- [11] Barbie, A., Hasselbring, W., & Hansen, M. (2024). Digital twin prototypes for supporting automated integration testing of smart farming applications. *Symmetry*, 16(2), 221. URL: <https://doi.org/10.3390/sym16020221>
- [12] Beautiful Soup Documentation. URL: <https://beautiful-soup-4.readthedocs.io/en/latest/> (visited on 2025-08-15).
- [13] Bernal, E., Wu, Q., Spiryagin, M., & Cole, C. (2024). Augmented digital twin for railway systems. *Vehicle System Dynamics*, 62(1), 67-83. URL: <https://doi.org/10.1080/00423114.2023.2194543>
- [14] Bertsch, C., Ahle, E., & Schulmeister, U. (2014, March). The Functional Mockup Interface-seen from an industrial perspective. In *Proceedings of the 10th International Modelica Conference (Vol. 96, p. 27)*. URL: <https://doi.org/10.3384/ECP1409627>
- [15] Bigelow, Stephen J.; Courtemanche, Meredith; Gills, Alexander S. "What Is DevOps? The Ultimate Guide". TechTarget. URL: <https://www.techtarget.com/searchitoperations/definition/DevOps> (visited on 2025-12-12).
- [16] Binkley, D. (2007). Source code analysis: A road map. *Future of Software Engineering (FOSE'07)*, 104-119. URL: <https://doi.org/10.1109/FOSE.2007.27>
- [17] Blochwitz, T., Otter, M., Arnold, M., Bausch, C., Clauß, C., Elmqvist, H., ... & Wolf, S. (2011, March). The functional mockup interface for tool independent exchange of simulation models. In *Proceedings of the 8th international Modelica conference (pp. 105-114)*. Linköping University Press. URL: <https://ep.liu.se/ecp/063/013/ecp11063013.pdf> (visited on 2025-08-15).

- [18] Botín-Sanabria, D. M., Mihaita, A. S., Peimbert-García, R. E., Ramírez-Moreno, M. A., Ramírez-Mendoza, R. A., & Lozoya-Santos, J. D. J. (2022). Digital twin technology challenges and applications: A comprehensive review. *Remote Sensing*, 14(6), 1335. URL: <https://doi.org/10.3390/rs14061335>
- [19] Brusa, E., Dagna, A., Delprete, C., & Gentile, R. (2023). An orchestration method for integrated multi-disciplinary simulation in digital twin applications. *Aerospace*, 10(7), 601. URL: <https://doi.org/10.3390/aerospace10070601>
- [20] Chacón, R., Ramonell, C., Posada, H., Sierra, P., Tomar, R., Martínez de la Rosa, C., ... & Wagmeister, S. (2024). Digital twinning during load tests of railway bridges-case study: the high-speed railway network, Extremadura, Spain. *Structure and Infrastructure Engineering*, 20(7-8), 1105-1119. URL: <https://doi.org/10.1080/15732479.2023.2264840>
- [21] Chandaluri, R., & Nelakuditi, U. R. (2024). Performance evaluation of electro-mechanical railway interlocking system for digital twin application. *Computers and Electrical Engineering*, 116, 109225. URL: <https://doi.org/10.1016/j.compeleceng.2024.109225>
- [22] Chen, Lianping. (2015). Continuous Delivery: Huge Benefits, but Challenges Too. *IEEE Software*. 32. DOI: 10.1109/MS.2015.27. URL: https://www.researchgate.net/publication/271635510_Continuous_Delivery_Huge_Benefits_but_Challenges_Too (visited on 2025-08-15).
- [23] Combemale, B., Jézéquel, J. M., Perez, Q., Vojtisek, D., Jansen, N., Michael, J., ... & Zhang, J. (2023, October). Model-based DevOps: foundations and challenges. In *2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)* (pp. 429-433). IEEE. URL: <https://doi.org/10.1109/MODELS-C59198.2023.00076>
- [24] Crespi, N., Drobot, A. T., & Minerva, R. (2023). The Digital Twin: What and Why?. In *The Digital Twin* (pp. 3-20). Cham: Springer International Publishing. URL: https://doi.org/10.1007/978-3-031-21343-4_1
- [25] DCP Specification 1.0 (pdf File). URL: https://dcp-standard.org/assets/DCP_Specification_v1.0.pdf (visited on 2025-08-15).
- [26] DCP Standard Homepage. URL: <https://dcp-standard.org/> (visited on 2025-08-15).

- [27] DCPLib software library on GitHub. URL: <https://github.com/modelica/DCPLib> (visited on 2025-08-15).
- [28] Deliverable D1.1.6 – Simulation Deployment Results M41 Area 1. URL: <https://www.rail4future.com/en/dam/jcr:88fc6d53-414f-43cc-9ff7-c366a43422f5/D-116-Simulation-Deployment-Results-M41.pdf> (visited on 2025-08-15).
- [29] Deliverable D1.3.5 – Report Implementation Strategies M36 Area 1. URL: <https://www.rail4future.com/en/dam/jcr:bc2b478b-2642-405a-a55b-0723dd9c70b7/D-135-Report-Implementation-Strategies-M36.pdf> (visited on 2025-08-15).
- [30] Dimitrova, E., & Tomov, S. (2021, September). Digital twins: an advanced technology for railways maintenance transformation. In 2021 13th Electrical Engineering Faculty Conference (BulEF) (pp. 1-5). IEEE. URL: <https://doi.org/10.1109/BulEF53491.2021.9690822>
- [31] Dittler, D., Frank, P., Hildebrandt, G., Peterson, L., Jazdi, N., & Weyrich, M. (2025). Model-Based Control for Power-to-X Platforms: Knowledge Integration for Digital Twins. arXiv preprint arXiv:2507.03553. URL: <https://doi.org/10.48550/arXiv.2507.03553>
- [32] Dobaj, J., Riel, A., Krug, T., Seidl, M., Macher, G., & Egretzberger, M. (2022, May). Towards digital twin-enabled DevOps for CPS providing architecture-based service adaptation & verification at runtime. In Proceedings of the 17th symposium on software engineering for adaptive and self-managing systems (pp. 132-143). URL: <https://doi.org/10.1145/3524844.3528057>
- [33] Dobaj, J., Riel, A., Macher, G., & Egretzberger, M. (2023). Towards devops for cyber-physical systems (cpss): Resilient self-adaptive software for sustainable human-centric smart cps facilitated by digital twins. *Machines*, 11(10), 973. URL: <https://doi.org/10.3390/machines11100973>
- [34] Durling, E., Palmkvist, E., & Henningsson, M. (2017, May). Fmi and ip protection of models: A survey of use cases and support in the standard. In *Modelica* (pp. 132-036). URL: <https://doi.org/10.3384/ecp17132329>
- [35] Enders, M. R., & Hoßbach, N. (2019). Dimensions of digital twin applications-a literature review. URL: <https://www.researchgate.net/profile/Martin-Enders->

[4/publication/359715537_Dimensions_of_Digital_Twin_Applications_-_A_Literature_Review/links/624ae8d921077329f2f1fb4f/Dimensions-of-Digital-Twin-Applications-A-Literature-Review.pdf](#) (visited on 2025-08-15).

[36] Erkoyuncu, J. A., Butala, P., & Roy, R. (2018). Digital twins: Understanding the added value of integrated models for through-life engineering services. *Procedia Manufacturing*, 16, 139-146. URL: <https://doi.org/10.1016/j.promfg.2018.10.167>

[37] Fitzgerald, J., Larsen, P. G., & Pierce, K. (2019). Multi-modelling and co-simulation in the engineering of cyber-physical systems: towards the digital twin. In *From Software Engineering to Formal Methods and Tools, and Back: Essays Dedicated to Stefania Gnesi on the Occasion of Her 65th Birthday* (pp. 40-55). Cham: Springer International Publishing. URL: https://doi.org/10.1007/978-3-030-30985-5_4

[38] FMI 2.0 Specification. URL: https://fmi-standard.org/assets/releases/FMI_for_ModelExchange_and_CoSimulation_v2.0.pdf (visited on 2025-08-15).

[39] FMI for Model Exchange 1.0 Specification (pdf File). URL: https://fmi-standard.org/assets/releases/FMI_for_ModelExchange_v1.0.pdf (visited on 2025-08-15).

[40] FMI Kit for Simulink Homepage on GitHub. URL: <https://github.com/CATIA-Systems/FMIKit-Simulink> (visited on 2024-04-01).

[41] FMI Standard Homepage. URL: <https://fmi-standard.org/> (visited on 2025-08-15).

[42] Freeman, E., & Forsgren, N. (2019). *DevOps. For Dummies*. URL: <https://learning.oreilly.com/library/view/devops-for-dummies/9781119552222/?ar=> (visited on 2025-08-17).

[43] Futai, M. M., Machado, L. B., Santos, R. R., Poncetti, B. L., Bittencourt, T. N., & Gamino, A. L. (2024). Digital twins for condition assessment of railway infrastructures. In *Digital railway infrastructure* (pp. 157-176). Cham: Springer Nature Switzerland. URL: https://doi.org/10.1007/978-3-031-49589-2_8

[44] Ghaboura, S., Ferdousi, R., Laamarti, F., Yang, C., & El Saddik, A. (2023). Digital twin for railway: A comprehensive survey. *IEEE Access*, 11, 120237-120257. URL: <https://doi.org/10.1109/ACCESS.2023.3327042>

- [45] Git Documentation. URL: <https://git-scm.com/docs/git> (visited on 2025-08-15).
- [46] GitHub webpage of the *cosim-cli* command-line co-simulation tool. URL: <https://github.com/open-simulation-platform/cosim-cli> (visited on 2025-08-15).
- [47] GitHub webpage of the *cosim-demo-app* application. URL: <https://github.com/open-simulation-platform/cosim-demo-app> (visited on 2025-08-15).
- [48] GitHub webpage of the *demo-cases* repository. URL: <https://github.com/open-simulation-platform/demo-cases> (visited on 2025-08-15).
- [49] GitHub webpage of the FMPy Python library. URL: <https://github.com/CATIA-Systems/FMPy> (visited on 2025-08-15).
- [50] Github webpage of the FMU Compliance Checker application. URL: <https://github.com/modelica-tools/FMUComplianceChecker> (visited on 2025-08-15).
- [51] GitHub webpage of the *libcosim* C++ library. URL: <https://github.com/open-simulation-platform/libcosim> (visited on 2025-08-15).
- [52] GitHub webpage of the *libcosimc* C library. URL: <https://github.com/open-simulation-platform/libcosimc> (visited on 2025-08-15).
- [53] GitHub webpage of the *osp-validator* Java application. URL: <https://github.com/open-simulation-platform/osp-validator> (visited on 2025-08-15).
- [54] GitHub webpage of the *proxy-fmu* C++ library. URL: <https://github.com/open-simulation-platform/proxy-fmu> (visited on 2025-08-15).
- [55] GitHub webpage of the PythonFMU Python library. URL: <https://github.com/NTNU-IHB/PythonFMU> (visited on 2025-08-15).
- [56] Hällqvist, R., Munjulury, R. C., Braun, R., Eek, M., & Krus, P. (2021, September). Engineering domain interoperability using the system structure and parameterization (ssp) standard. In Modelica Conferences (pp. 37-48). URL: <https://doi.org/10.3384/ecp2118137>
- [57] Hamarat, M., Papaelias, M., & Kaewunruen, S. (2022). Fatigue damage assessment of complex railway turnout crossings via Peridynamics-based

digital twin. Scientific reports, 12(1), 14377. URL:

<https://doi.org/10.1038/s41598-022-18452-w>

[58] Hatledal, L. I., & Fagerhaug, E. (2022). Enhancing SSP creation using sspgen. In Modelica Conferences (pp. 115-119). URL:

<https://doi.org/10.3384/ECP21186115>

[59] Hatledal, L. I., Skulstad, R., Li, G., Styve, A., & Zhang, H. (2020). Co-simulation as a fundamental technology for twin ships. URL:

<https://dx.doi.org/10.4173/mic.2020.4.2> (visited on 2025-08-15).

[60] Hatledal, L. I., Styve, A., Hovland, G., & Zhang, H. (2019). A language and platform independent co-simulation framework based on the functional mock-up interface. IEEE Access, 7, 109328-109339. URL:

<https://doi.org/10.1109/ACCESS.2019.2933275>

[61] Havard, V., Jeanne, B., Lacomblez, M., & Baudry, D. (2019). Digital twin and virtual reality: a co-simulation environment for design and assessment of industrial workstations. Production & Manufacturing Research, 7(1), 472-489. URL:

<https://doi.org/10.1080/21693277.2019.1660283>

[62] Hernández, L. A., & Hernández, S. (1997). Application of digital 3D models on urban planning and highway design. WIT Transactions on the built environment, 33. URL: <https://www.witpress.com/elibrary/wit-transactions-on-the-built-environment/33/8418> (visited on 2025-08-15).

[63] Hodson, T. O. (2022). Root mean square error (RMSE) or mean absolute error (MAE): When to use them or not. Geoscientific Model Development Discussions, 2022, 1-10. URL: <https://doi.org/10.5194/gmdd-7-1525-2014>

[64] Hugues, J., Hristosov, A., Hudak, J. J., & Yankel, J. (2020, October). TwinOps-DevOps meets model-based engineering and digital twins for the engineering of CPS. In Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings (pp. 1-5). URL:

<https://doi.org/10.1145/3417990.3421446>

[65] Infante, S., Martín, C., Robles, J., Rubio, B., Díaz, M., Perea, R. G., ... & Poyato, E. C. (2024). Integrating FMI and ML/AI models on the open-source digital twin framework OpenTwins. Software: Practice and Experience, 54(8), 1470-1490. URL: <https://doi.org/10.1002/spe.3322>

- [66] Iwnicki, S. (Ed.). (1999). The Manchester Benchmarks for Rail Vehicle Simulation (1st ed.). Routledge. URL: <https://doi.org/10.1201/9780203736425>
- [67] Jones, D., Nassehi, A., Snider, C., Gopsill, J., Rosso, P., Real, R., ... & Hicks, B. (2021). Towards integrated version control of virtual and physical artefacts in new product development: inspirations from software engineering and the digital twin paradigm. *Procedia CIRP*, 100, 283-288. URL: <https://doi.org/10.1016/j.procir.2021.05.121>
- [68] Kaewunruen, S., & Lian, Q. (2019). Digital twin aided sustainability-based lifecycle management for railway turnout systems. *Journal of Cleaner Production*, 228, 1537-1551. URL: <https://doi.org/10.1016/j.jclepro.2019.04.156>
- [69] Kaewunruen, S., & Xu, N. (2018). Digital twin for sustainability evaluation of railway station buildings. *Frontiers in Built Environment*, 4, 77. URL: <https://doi.org/10.3389/fbuil.2018.00077>
- [70] Kaewunruen, S., AbdelHadi, M., Kongpuang, M., Pansuk, W., & Remennikov, A. M. (2022). Digital twins for managing railway bridge maintenance, resilience, and climate change adaptation. *Sensors*, 23(1), 252. URL: <https://doi.org/10.3390/s23010252>
- [71] Kampczyk, A., & Dybeł, K. (2021). The fundamental approach of the digital twin application in railway turnouts with innovative monitoring of weather conditions. *Sensors*, 21(17), 5757. URL: <https://doi.org/10.3390/s21175757>
- [72] Kern, D. (2023). Leveraging semantic technologies for the application in multi-domain digital twins [Diploma Thesis, Technische Universität Wien]. *repositUM*. URL: <https://doi.org/10.34726/hss.2023.99482>
- [73] Kothari, C. R. (2004). *Research methodology: Methods and techniques*. New Age International. URL: <https://dspace.unitywomenscollege.ac.in/bitstream/123456789/163/1/Research%20Methodology%20C%20R%20Kothari.pdf> (visited on 2025-08-15).
- [74] Krammer, M., & Benedikt, M. (2018, September). Configuration of slaves based on the distributed co-simulation protocol. In *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)* (Vol. 1, pp. 195-202). IEEE. URL: <https://doi.org/10.1109/ETFA.2018.8502564>

- [75] Krammer, M., Benedikt, M., Blochwitz, T., Alekeish, K., Amringer, N., Kater, C., ... & Andert, J. (2018, July). The distributed co-simulation protocol for the integration of real-time systems and simulation environments. In Proceedings of the 50th Computer Simulation Conference (pp. 1-14). URL: <https://dl.acm.org/doi/10.5555/3275382.3275383> (visited on 2025-08-15).
- [76] Kugu, O., Zhou, S., Nowak, R., Müller, G., Reiterer, S.H., Meierhofer, A., Wurth, L. & Grafinger, M. (2023, December), An FMI-and SSP-based Model Integration Methodology for a Digital Twin Platform of a Holistic Railway Infrastructure System. In Modelica Conferences (pp. 717-726). URL: <https://doi.org/10.3384/ecp204717>
- [77] Kugu, O., Zhou, S., Reiterer, S.H., Schwaiger, M., Wurth, L. and Grafinger, M. (2024), Pipeline-based Automated Integration and Delivery Testing of Simulation Assets with FMI/SSP in a Railway Digital Twin. American Modelica Conference 2024. URL: <https://doi.org/10.3384/ecp207189>
- [78] Larsen, P. G., Esterle, L., Fitzgerald, J., & Frasher, M. (2023). Fault injection in co-simulation and digital twins for cyber-physical robotic systems. In Applicable formal methods for safe industrial products: essays dedicated to Jan Peleska on the occasion of his 65th birthday (pp. 222-236). Cham: Springer Nature Switzerland. URL: https://doi.org/10.1007/978-3-031-40132-9_14
- [79] Lazoglu, A., Naraniecki, H., Zaidman, I., & Marx, S. (2023). A monitoring based digital twin for the Filstal bridges. In Life-cycle of structures and infrastructure systems (pp. 205-212). CRC Press. URL: <https://doi.org/10.1201/9781003323020-22>
- [80] Lott, S.F. and Phillips, D. (2021) Python Object-Oriented Programming: Build Robust and Maintainable Object-Oriented Python Applications and Libraries. Fourth edition. Birmingham: Packt Publishing, Limited. URL: <https://books.google.at/books?hl=de&lr=&id=J5Y2EAAAQBAJ&oi=fnd&pg=PP1&dq=Python+Object-Oriented+Programming:+Build+Robust+and+Maintainable+Object-Oriented+Python+Application&ots=j78cJPGcdY&sig=QxSnTKib1SgUqOM0dejVPUBKUu8#v=onepage&q=Python%20Object-Oriented%20Programming%3A%20Build%20Robust%20and%20Maintainable>

[%20Object-Oriented%20Python%20Application&f=false](#) (visited on 2025-08-15).

[81] Maritime reference models of OSP. URL: <https://open-simulation-platform.github.io/demo-cases> (visited on 2025-08-15).

[82] Markusfeld, D. (2024). Automating build, deployment, and monitoring of model-based digital twins (Doctoral dissertation, Technische Universität Wien). URL: <https://doi.org/10.34726/hss.2024.92044>

[83] Matplotlib Python library. URL: <https://matplotlib.org/> (visited on 2025-08-15).

[84] Mertens, J., & Denil, J. (2021, August). The digital twin as a common knowledge base in devops to support continuous system evolution. In International Conference on Computer Safety, Reliability, and Security (pp. 158-170). Cham: Springer International Publishing. URL: https://doi.org/10.1007/978-3-030-83906-2_12

[85] Miao, R., Liu, S., Sun, Y., Du, M., & Bao, J. (2025). Streamlining digital twin development and operation with DTOps. Journal of Intelligent Manufacturing, 1-20. URL: <https://doi.org/10.1007/s10845-025-02608-2>

[86] Milošević, M. D., Pålsson, B. A., Nissen, A., Nielsen, J. C., & Johansson, H. (2021, August). Demonstration of a digital twin framework for model-based operational condition monitoring of crossing panels. In The IAVSD International Symposium on Dynamics of Vehicles on Roads and Tracks (pp. 95-105). Cham: Springer International Publishing. URL: https://doi.org/10.1007/978-3-031-07305-2_11

[87] Mohseni, S. R., Zeitouni, M. J., Parvaresh, A., Abrazeh, S., Gheisarnejad, M., & Khooban, M. H. (2023). FMI real-time co-simulation-based machine deep learning control of HVAC systems in smart buildings: Digital-twins technology. Transactions of the Institute of Measurement and Control, 45(4), 661-673. URL: <https://doi.org/10.1177/01423312221119635>

[88] Nandgaonkar, S., & Khatavkar, V. (2022, October). CI-CD pipeline for content releases. In 2022 IEEE 3rd Global Conference for Advancement in Technology (GCAT) (pp. 1-4). IEEE. URL: <https://doi.org/10.1109/GCAT55367.2022.9972129>

- [89] Naraniecki, H., Lazoglu, A., Marx, S., & Zaidman, I. (2023). Concept for a digital twin of railway bridges on the example of the new Filstal bridges. *ce/papers*, 6(5), 711-717. URL: <https://doi.org/10.1002/cepa.2050>
- [90] Neelamraju, S. (2024). Towards a standardized framework for collaborative ship powertrain design using the system structure and parameterization standard. URL: <https://urn.fi/URN:NBN:fi:aalto-202405263719> (visited on 2025-07-30).
- [91] Negri, E., Fumagalli, L., Cimino, C., & Macchi, M. (2019). FMU-supported simulation for CPS digital twin. *Procedia manufacturing*, 28, 201-206. URL: <https://doi.org/10.1016/j.promfg.2018.12.033>
- [92] Nhamage, I. A., Dang, N. S., Horas, C. S., Poças Martins, J., Matos, J. A., & Calçada, R. (2023). Performing fatigue state characterization in railway steel bridges using digital twin models. *Applied Sciences*, 13(11), 6741. URL: <https://doi.org/10.3390/app13116741>
- [93] Ochel, L. A., Braun, R., Thiele, B., Asghar, A., Buffoni, L., Eek, M., ... & Sjölund, M. (2019, March). OMSimulator-Integrated FMI and TLM-based Co-simulation with Composite Model Editing and SSP. In *Modelica* (pp. 157-007). URL: <https://www.doi.org/10.3384/ecp1915769>
- [94] OMSimulator Homepage. URL: <https://openmodelica.org/free-and-open-source-software/omsimulator/> (visited on 2025-08-15).
- [95] OpenMCX Homepage on Github. URL: <https://github.com/eclipse-openmcx/openmcx> (visited on 2025-08-15).
- [96] Os Python library. URL: <https://docs.python.org/3/library/os.html> (visited on 2025-08-15).
- [97] OSP Homepage. URL: <https://opensimulationplatform.com/> (visited on 2025-08-15).
- [98] OSP Interface Specification. URL: <https://opensimulationplatform.com/specification/> (visited on 2025-08-15).
- [99] OSP Webpage – Background. URL: <https://opensimulationplatform.com/background/>. (visited on 2025-08-15).
- [100] Padovano, A., Longo, F., Manca, L., & Grugni, R. (2024). Improving safety management in railway stations through a simulation-based digital twin

- approach. *Computers & Industrial Engineering*, 187, 109839. URL: <https://doi.org/10.1016/j.cie.2023.109839>
- [101] Palmieri, M., Quadri, C., Fagiolini, A., & Bernardeschi, C. (2023). Co-simulated digital twin on the network edge: A vehicle platoon. *Computer Communications*, 212, 35-47. URL: <https://doi.org/10.1016/j.comcom.2023.09.019>
- [102] Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of management information systems*, 24(3), 45-77. URL: <https://doi.org/10.2753/MIS0742-1222240302>
- [103] Perabo, F., Park, D., Zadeh, M. K., Smogeli, Ø., & Jamt, L. (2020, June). Digital twin modelling of ship power and propulsion systems: Application of the open simulation platform (osp). In *2020 IEEE 29th International Symposium on Industrial Electronics (ISIE)* (pp. 1265-1270). IEEE. URL: <https://doi.org/10.1109/ISIE45063.2020.9152218>
- [104] Pfeifer, D., Baumann, A., Giani, M., Scheifele, C., & Fehr, J. (2023). Hybrid digital twins using FMUs to increase the validity and domain of virtual commissioning simulations. In *Stuttgart Conference on Automotive Production* (pp. 200-209). Cham: Springer International Publishing. URL: https://doi.org/10.1007/978-3-031-27933-1_19
- [105] Python webpage of the csv Python library. URL: <https://docs.python.org/3/library/csv.html> (visited on 2025-08-15).
- [106] Python webpage of the json Python library. URL: <https://docs.python.org/3/library/json.html> (visited on 2025-08-15).
- [107] Rail4Future Deliverables of Area 1: Simulation Platform for Railway Systems. URL: <https://www.rail4future.com/en/insights-rd/r4f-area1> (visited on 2025-08-15).
- [108] Rail4Future project homepage. URL: <https://www.rail4future.com/> (visited on 2025-08-15).
- [109] Rayanagoudar, S. F., Hampannavar, P. S., Pujari, J. D., & Parvati, V. K. (2018). Enhancement of CI/CD Pipelines with Jenkins BlueOcean. *International Journal of Computer Science and Engineering*. URL: <https://doi.org/10.26438/ijcse/v6i6.10481053>

- [110] Reiterer, S. H. (2025). Continuous integration and continuous deployment for cyber-physical systems by graph based meta data models [Dissertation, Technische Universität Graz]. TU Graz Repository. URL: <https://doi.org/10.3217/kd4cn-88v78>
- [111] Reiterer, S. H., Schiffer, C., & Benedikt, M. (2022). A Graph-Based Metadata Model for DevOps in Simulation-Driven Development and Generation of DCP Configurations. *Electronics*, 11(20), 3325. URL: <https://doi.org/10.3390/electronics11203325>
- [112] Reiterer, S. H., Schiffer, C., & Schwaiger, M. (2023, December). A graph-based meta-data model for devops: Extensions to ssp and sysml2 and a review on the dcp standard. In *Modelica Conferences* (pp. 159-166). URL: <https://doi.org/10.3384/ecp204159>
- [113] Rindarøy, M., Nordahl, H., Sadjina, S., Skjong, S., & Hagaseth, M. (2025). Adding higher-level semantics to Functional Mock-up Units for easier, faster, and more robust co-simulation connections. *Software and Systems Modeling*, 24(2), 471-487. URL: <https://doi.org/10.1007/s10270-024-01244-3>
- [114] Sarp, S., Kuzlu, M., Jovanovic, V., Polat, Z., & Guler, O. (2024). Digitalization of railway transportation through AI-powered services: digital twin trains. *European Transport Research Review*, 16(1), 58. URL: <https://doi.org/10.1186/s12544-024-00679-5>
- [115] Scheepers, M. J. (2014, June). Virtualization and containerization of application infrastructure: A comparison. In *21st twente student conference on IT* (Vol. 21, pp. 1-7).
- [116] Scheifele, C., Verl, A., & Riedel, O. (2019). Real-time co-simulation for the virtual commissioning of production systems. *Procedia CIRP*, 79, 397-402. URL: <https://doi.org/10.1016/j.procir.2019.02.104>
- [117] Scherma, V. (2024). Design and implementation of an integrated DevOps framework for Digital Twins as a Service software platform (Doctoral dissertation, Politecnico di Torino). URL: <https://webthesis.biblio.polito.it/id/eprint/34033> (visited on 2025-08-15).
- [118] Schluse, M., & Rossmann, J. (2016, October). From simulation to experimentable digital twins: Simulation-based development and operation of complex technical systems. In *2016 IEEE international symposium on*

systems engineering (ISSE) (pp. 1-6). IEEE. URL:

<https://doi.org/10.1109/SysEng.2016.7753162>

[119] Schluse, M., Atorf, L., & Rossmann, J. (2017, April). Experimentable digital twins for model-based systems engineering and simulation-based development. In 2017 annual IEEE international systems conference (syscon) (pp. 1-8). IEEE. URL: <https://doi.org/10.1109/SYSCON.2017.7934796>

[120] Segura, M., Poggi, T., & Barcena, R. (2023). A generic interface for x-in-the-loop simulations based on distributed co-simulation protocol. IEEE Access, 11, 5578-5595. DOI: <https://doi.org/10.1109/ACCESS.2023.3237075>

[121] Semeraro, C., Lezoche, M., Panetto, H., & Dassisti, M. (2021). Digital twin paradigm: A systematic literature review. Computers in Industry, 130, 103469. URL: <https://doi.org/10.1016/j.compind.2021.103469>

[122] Sepasgozar, S. M. (2021). Differentiating digital twin from digital shadow: Elucidating a paradigm shift to expedite a smart, sustainable built environment. Buildings, 11(4), 151. URL: <https://doi.org/10.3390/buildings11040151>

[123] Shabelnikov, A. N., & Olgezyer, I. A. (2019, December). Technology and mathematical basis of digital twin creation in railway infrastructure. In International Conference on Intelligent Information Technologies for Industry (pp. 688-695). Cham: Springer International Publishing. URL: https://doi.org/10.1007/978-3-030-50097-9_70

[124] Shah, N. H., Le Henaff, P., Schiffer, C., Krammer, M., & Benedikt, M. (2021, September). Accurate robot simulation for industrial manufacturing processes using fmi and dcp standards. In Modelica Conferences (pp. 673-679). URL: <https://doi.org/10.3384/ecp21181673>

[125] Shcherbakov, M. V., Brebels, A., Shcherbakova, N. L., Tyukov, A. P., Janovsky, T. A., & Kamaev, V. A. E. (2013). A survey of forecast error measures. World applied sciences journal, 24(24), 171-176. URL: [https://www.idosi.org/wasj/wasj\(ITMIES\)13/28.pdf](https://www.idosi.org/wasj/wasj(ITMIES)13/28.pdf)

[126] Shen, C., Zhang, P., Dollevoet, R., Zoeteman, A., & Li, Z. (2023). Evaluating railway track stiffness using axle box accelerations: a digital twin approach. Mechanical Systems and Signal Processing, 204, 110730. URL: <https://doi.org/10.1016/j.ymssp.2023.110730>

- [127] Smith, J.E. and Nair, R. (2005) Virtual Machines. Morgan Kaufmann. Available at: <https://doi.org/10.1016/B978-1-55860-910-5.X5000-9>
- [128] SSP Specification 1.0 (pdf-file). URL: <https://ssp-standard.org/publications/SSP10/SystemStructureAndParameterization10.pdf> (visited on 2025-08-15).
- [129] SSP Standard Homepage. URL: <https://ssp-standard.org/> (visited on 2025-08-15).
- [130] Suárez-Figueroa, M. C., Gómez-Pérez, A., & Fernández-López, M. (2011). The NeOn methodology for ontology engineering. In *Ontology engineering in a networked world* (pp. 9-34). Berlin, Heidelberg: Springer Berlin Heidelberg. URL: https://doi.org/10.1007/978-3-642-24794-1_2
- [131] Swartout, P., Verona, Joakim; Duffy, Michael; Swartout, Paul and Duffy, M. (2016) 'Containerization with Docker', in *Learning DevOps: Continuously Deliver Better Software*. United Kingdom: Packt Publishing, Limited. URL: <https://learning.oreilly.com/library/view/learning-devops-continuously/9781787126619/?ar=> (visited on 2025-08-15).
- [132] Tools supporting the DCP standard. URL: <https://dcp-standard.org/tools/> (visited on 2025-08-15).
- [133] Tools supporting the FMI standard. URL: <https://fmi-standard.org/tools/> (visited on 2025-08-15).
- [134] Tools supporting the SSP standard. URL: <https://ssp-standard.org/tools/> (visited on 2025-08-15).
- [135] Turco, L., Zhao, J., Xu, Y., & Tsourdos, A. (2024, March). A study on co-simulation digital twin with MATLAB and AirSim for future advanced air mobility. In *2024 IEEE Aerospace Conference* (pp. 1-18). IEEE. URL: <https://doi.org/10.1109/AERO58975.2024.10521333>
- [136] Ugarte Querejeta, M., Etxeberria, L., & Sagardui, G. (2020, September). Towards a devops approach in cyber physical production systems using digital twins. In *International Conference on Computer Safety, Reliability, and Security* (pp. 205-216). Cham: Springer International Publishing. URL: https://doi.org/10.1007/978-3-030-55583-2_15
- [137] Vico Homepage on GitHub. URL: <https://github.com/NTNU-IHB/Vico> (visited on 2025-08-15).

- [138] Wadhams, Z., Reinhold, A. M., & Izurieta, C. (2024, March). Automating Static Code Analysis Through CI/CD Pipeline Integration. In 2nd International Workshop on Mining Software Repositories for Privacy and Security, MSR4P&S,(SANER 2024), Rovaniemi, Finland. URL: <https://doi.org/10.1109/SANER-C62648.2024.00021>
- [139] Wen, Z., Zhao, J., Xu, Y., & Tsourdos, A. (2024, March). A co-simulation digital twin with SUMO and AirSim for testing lane-based UTM system concept. In 2024 IEEE Aerospace Conference (pp. 1-11). IEEE. URL: <https://doi.org/10.1109/AERO58975.2024.10521156>
- [140] Wiens, M., Meyer, T., & Thomas, P. (2021, September). The potential of FMI for the development of digital twins for large modular multi-domain systems. In Modelica Conferences (pp. 235-240). URL: <https://doi.org/10.3384/ecp21181235>
- [141] Wurth, L. (2025). Traction Control System for Co-Simulation in a Digital Twin Platform [Diploma Thesis, Technische Universität Wien]. URL: <https://doi.org/10.34726/hss.2025.110720>
- [142] Zampetti, F., Scalabrino, S., Oliveto, R., Canfora, G., & Di Penta, M. (2017, May). How open source projects use static code analysis tools in continuous integration pipelines. In 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR) (pp. 334-344). IEEE. URL: <https://doi.org/10.1109/MSR.2017.2>
- [143] Zampetti, F., Tamburri, D., Panichella, S., Panichella, A., Canfora, G., & Di Penta, M. (2023). Continuous integration and delivery practices for cyber-physical systems: An interview-based study. *ACM Transactions on Software Engineering and Methodology*, 32(3), 1-44. URL: <https://doi.org/10.1145/3571854>
- [144] Zhao, J., Conrad, C., Fremond, R., Mukherjee, A., Delezenne, Q., Su, Y., ... & Tsourdos, A. (2023, October). Co-simulation digital twin framework for testing future advanced air mobility concepts: A study with BlueSky and AirSim. In 2023 IEEE/AIAA 42nd Digital Avionics Systems Conference (DASC) (pp. 1-10). IEEE. URL: <https://doi.org/10.1109/DASC58513.2023.10311124>
- [145] Zhou, C., Gao, L., Cai, X., Ding, H., Li, K., & Li, W. (2024). Digital twin technology for continuously welded turnout on high-speed railway bridges

- based on improved MOPSO algorithm. *Structural and Multidisciplinary Optimization*, 67(6), 87. URL: <https://doi.org/10.1007/s00158-024-03806-x>
- [146] Zhou, N., Zhou, H. and Hoppe, D. (2023) 'Containerization for High Performance Computing Systems: Survey and Prospects', *IEEE transactions on software engineering*, 49(4), pp. 2722–2740. URL: <https://doi.org/10.1109/TSE.2022.3229221>
- [147] Zhou, S., Dumss, S., Nowak, R., Riegler, R., Kugu, O., Krammer, M. & Grafinger, M. (2022), A Conceptual Model-based Digital Twin Platform for Holistic Large-scale Railway Infrastructure Systems. *CIRP Design 2022 (Vol. 109, pp. 362-367)*. URL: <https://doi.org/10.1016/j.procir.2022.05.263>
- [148] Zhou, S., Kugu, O., Reiterer, S.H., Wurth, L. & Grafinger, M. (2024), A Reinforcement-Learning-based Parameter Tuning Methodology for Traction Control in the Holistic Railway Digital Twin System. In *CIRP Design 2024 (Vol. 128, pp. 828-833)*. URL: <https://doi.org/10.1016/j.procir.2024.06.040>
- [149] Zhou, S., Meierhofer, A., Kugu, O., Xia, Y. & Grafinger, M. (2023), A Machine-Learning-based Surrogate Modeling Methodology for Submodel Integration in the Holistic Railway Digital Twin Platform. *CIRP Design 2023 (Vol. 119, pp. 345-350)*. URL: <https://doi.org/10.1016/j.procir.2023.02.141>
- [150] Zigart, T. (2022). *Entwicklung eines multikriteriellen Evaluierungsmodells für industrielle Assistenzsysteme (Doctoral dissertation, Technische Universität Wien)*. URL: <https://doi.org/10.34726/hss.2022.98440>
- [151] Zipfile Python library. URL: <https://docs.python.org/3/library/zipfile.html> (visited on 2025-08-15).

Appendix

A.1 Description Files

Appendix 1: An example model description XML file belonging to the FMI standard.

```
<fmiModelDescription fmiVersion="2.0" modelName="RLTB" guid="4a2271
09-0b1a-11f0-ab4f-c03c59042661" generationTool="PythonFMU
0.6.2" generationDateAndTime="2025-03-
27T14:46:40+00:00" variableNamingConvention="structured" descriptio
n="Rail4Future - Dummy Usecase Fatigue Assessment as input to Area
1" author="ozankugu">
<CoSimulation needsExecutionTool="true" canHandleVariableCommunicat
ionStepSize="true" canInterpolateInputs="false" canBeInstantiatedOn
lyOncePerProcess="false" canGetAndSetFMUstate="false" canSerializeF
MUstate="false" modelIdentifier="RLTB" canNotUseMemoryManagementFun
ctions="true">
<SourceFiles>
<File name="headers/cppfmu/cppfmu_common.hpp"/>
<File name="headers/cppfmu/cppfmu_cs.hpp"/>
<File name="headers/pythonfmu/PySlaveInstance.hpp"/>
<File name="headers/pythonfmu/PyState.hpp"/>
<File name="cpp/cppfmu_cs.cpp"/>
<File name="cpp/fmi_functions.cpp"/>
<File name="cpp/PySlaveInstance.cpp"/>
<File name="CMakeLists.txt"/>
</SourceFiles>
</CoSimulation>
<LogCategories>
<Category name="logStatusWarning" description="Log messages with
fmi2Warning status."/>
<Category name="logStatusDiscard" description="Log messages with
fmi2Discard status."/>
<Category name="logStatusError" description="Log messages with
fmi2Error status."/>
<Category name="logStatusFatal" description="Log messages with
fmi2Fatal status."/>
<Category name="logAll" description="Log all messages."/>
</LogCategories>
<ModelVariables>
<ScalarVariable name="inputfolder" valueReference="0" causality="pa
rameter" variability="fixed" initial="exact">
<String start="input_data_2"/>
</ScalarVariable>
<ScalarVariable name="outputfolder" valueReference="1" causality="p
arameter" variability="fixed" initial="exact">
<String start="output_data_2"/>
</ScalarVariable>
</ModelVariables>
<ModelStructure/>
</fmiModelDescription>
```

Appendix 2: An example XML-based SSD file belonging to the SSP standard.

```

<?xml version='1.0' encoding='UTF-8'?>
<ssd:SystemStructureDescription xmlns:ssc="http://ssp-
standard.org/SSP1/SystemStructureCommon" xmlns:ssd="http://ssp-
standard.org/SSP1/SystemStructureDescription"
xmlns:ssb="http://ssp-
standard.org/SSP1/SystemStructureSignalDictionary"
xmlns:ssv="http://ssp-
standard.org/SSP1/SystemStructureParameterValues"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.0"
name="231121_AntiSlip_MC" generationTool="Model.CONNECT"
generationDateAndTime="2023-11-21T11:49:28Z"
xsi:schemaLocation="http://ssp-
standard.org/SSP1/SystemStructureDescription http://ssp-
standard.org/SSP1/SSP/1.0/SystemStructureDescription.xsd">
  <ssd:System name="231121_AntiSlip_MC">
    <ssd:ElementGeometry x1="-21.0" y1="24.0" x2="-9.0" y2="36.0"/>
    <ssd:Elements>
      <ssd:Component name="Simpack_MBS_FMU" type="application/x-
fmu-sharedlibrary"
source="resources/ViF_ErIk_ManBM_v01_ORELow_FMI2_Linux.fmu">
        <ssd:Connectors>
          <ssd:Connector name="$UI_Controller_1" kind="input">
            <ssc:Real/>
            <ssd:ConnectorGeometry x="0.0" y="0.875"/>
          </ssd:Connector>
          <ssd:Connector name="$UI_Controller_2" kind="input">
            <ssc:Real/>
            <ssd:ConnectorGeometry x="0.0" y="0.75"/>
          </ssd:Connector>
          <ssd:Connector name="$UI_Controller_3" kind="input">
            <ssc:Real/>
            <ssd:ConnectorGeometry x="0.0" y="0.625"/>
          </ssd:Connector>
          <ssd:Connector name="$UI_Controller_4" kind="input">
            <ssc:Real/>
            <ssd:ConnectorGeometry x="0.0" y="0.5"/>
          </ssd:Connector>
          <ssd:Connector name="$UI_Vehicle_Position_desired"
kind="input">
            <ssc:Real/>
            <ssd:ConnectorGeometry x="0.0" y="0.375"/>
          </ssd:Connector>
          <ssd:Connector name="$UI_Vehicle_Speed_desired"
kind="input">
            <ssc:Real/>
            <ssd:ConnectorGeometry x="0.0" y="0.25"/>
          </ssd:Connector>
          <ssd:Connector name="$UI_Vehicle_Acceleration_desired"
kind="input">
            <ssc:Real/>
            <ssd:ConnectorGeometry x="0.0" y="0.125"/>
          </ssd:Connector>
          <ssd:Connector name="$Y_BG01_F_PS_11" kind="output">
            <ssc:Real/>
            <ssd:ConnectorGeometry x="1.0" y="0.9411765"/>
        </ssd:Connectors>
      </ssd:Component>
    </ssd:Elements>
  </ssd:System>
</ssd:SystemStructureDescription>

```

```

    <ssd:ConnectorGeometry x="1.0" y="0.9411765"/>
  </ssd:Connector>
  <ssd:Connector name="$Y_BG01_F_PS_12" kind="output">
    <ssc:Real/>
    <ssd:ConnectorGeometry x="1.0" y="0.88235295"/>
  </ssd:Connector>
  <ssd:Connector name="$Y_BG01_F_PS_21" kind="output">
    <ssc:Real/>
    <ssd:ConnectorGeometry x="1.0" y="0.8235294"/>
  </ssd:Connector>
  <ssd:Connector name="$Y_BG01_F_PS_22" kind="output">
    <ssc:Real/>
    <ssd:ConnectorGeometry x="1.0" y="0.7647059"/>
  </ssd:Connector>
  <ssd:Connector name="$Y_BG02_F_PS_11" kind="output">
    <ssc:Real/>
    <ssd:ConnectorGeometry x="1.0" y="0.7058824"/>
  </ssd:Connector>
  <ssd:Connector name="$Y_BG02_F_PS_12" kind="output">
    <ssc:Real/>
    <ssd:ConnectorGeometry x="1.0" y="0.64705884"/>
  </ssd:Connector>
  <ssd:Connector name="$Y_BG02_F_PS_21" kind="output">
    <ssc:Real/>
    <ssd:ConnectorGeometry x="1.0" y="0.5882353"/>
  </ssd:Connector>
  <ssd:Connector name="$Y_BG02_F_PS_22" kind="output">
    <ssc:Real/>
    <ssd:ConnectorGeometry x="1.0" y="0.5294118"/>
  </ssd:Connector>
  <ssd:Connector name="$Y_TrackPos" kind="output">
    <ssc:Real/>
    <ssd:ConnectorGeometry x="1.0" y="0.47058824"/>
  </ssd:Connector>
  <ssd:Connector name="$Y_VehicleSpeed" kind="output">
    <ssc:Real/>
    <ssd:ConnectorGeometry x="1.0" y="0.4117647"/>
  </ssd:Connector>
  <ssd:Connector name="$Y_Excitation_z1" kind="output">
    <ssc:Real/>
    <ssd:ConnectorGeometry x="1.0" y="0.3529412"/>
  </ssd:Connector>
  <ssd:Connector name="$Y_Excitation_z2" kind="output">
    <ssc:Real/>
    <ssd:ConnectorGeometry x="1.0" y="0.29411766"/>
  </ssd:Connector>
  <ssd:Connector name="$Y_w_BG01_RS1" kind="output">
    <ssc:Real/>
    <ssd:ConnectorGeometry x="1.0" y="0.23529412"/>
  </ssd:Connector>
  <ssd:Connector name="$Y_w_BG01_RS2" kind="output">
    <ssc:Real/>
    <ssd:ConnectorGeometry x="1.0" y="0.1764706"/>
  </ssd:Connector>
  <ssd:Connector name="$Y_w_BG02_RS1" kind="output">
    <ssc:Real/>
    <ssd:ConnectorGeometry x="1.0" y="0.11764706"/>
  </ssd:Connector>
  <ssd:Connector name="$Y_w_BG02_RS2" kind="output">
    <ssc:Real/>

```

```

<ssd:Connector name="$Y_w_BG02_RS1" kind="output">
  <ssc:Real/>
  <ssd:ConnectorGeometry x="1.0" y="0.11764706"/>
</ssd:Connector>
<ssd:Connector name="$Y_w_BG02_RS2" kind="output">
  <ssc:Real/>
  <ssd:ConnectorGeometry x="1.0" y="0.05882353"/>
</ssd:Connector>
</ssd:Connectors>
<ssd:ElementGeometry x1="109.62603851189837" y1="-
39.23634132121461" x2="218.7547290990208" y2="69.89235283807642"/>
<ssd:ParameterBindings>
  <ssd:ParameterBinding type="application/x-ssp-parameter-
set" source="resources/Simpack_MBS_FMU.ssv"/>
</ssd:ParameterBindings>
</ssd:Component>
<ssd:Component name="PID_Controller_Simulink_FMU"
type="application/x-fmu-sharedlibrary"
source="resources/PID_controller_Simulink_FMU.fmu">
  <ssd:Connectors>
    <ssd:Connector name="$Y_VehicleSpeed" kind="input">
      <ssc:Real/>
      <ssd:ConnectorGeometry x="0.0" y="0.8333333"/>
    </ssd:Connector>
    <ssd:Connector name="$Y_w_BG01_RS1" kind="input">
      <ssc:Real/>
      <ssd:ConnectorGeometry x="0.0" y="0.6666667"/>
    </ssd:Connector>
    <ssd:Connector name="$Y_w_BG01_RS2" kind="input">
      <ssc:Real/>
      <ssd:ConnectorGeometry x="0.0" y="0.5"/>
    </ssd:Connector>
    <ssd:Connector name="$Y_w_BG02_RS1" kind="input">
      <ssc:Real/>
      <ssd:ConnectorGeometry x="0.0" y="0.33333334"/>
    </ssd:Connector>
    <ssd:Connector name="$Y_w_BG02_RS2" kind="input">
      <ssc:Real/>
      <ssd:ConnectorGeometry x="0.0" y="0.16666667"/>
    </ssd:Connector>
    <ssd:Connector name="$UI_Controller_1" kind="output">
      <ssc:Real/>
      <ssd:ConnectorGeometry x="1.0" y="0.875"/>
    </ssd:Connector>
    <ssd:Connector name="$UI_Controller_2" kind="output">
      <ssc:Real/>
      <ssd:ConnectorGeometry x="1.0" y="0.75"/>
    </ssd:Connector>
    <ssd:Connector name="$UI_Controller_3" kind="output">
      <ssc:Real/>
      <ssd:ConnectorGeometry x="1.0" y="0.625"/>
    </ssd:Connector>
    <ssd:Connector name="$UI_Controller_4" kind="output">
      <ssc:Real/>
      <ssd:ConnectorGeometry x="1.0" y="0.5"/>
    </ssd:Connector>
    <ssd:Connector name="$UI_Vehicle_Position_desired"
kind="output">
      <ssc:Real/>

```

```

        <ssd:ConnectorGeometry x="1.0" y="0.375"/>
    </ssd:Connector>
    <ssd:Connector name="$UI_Vehicle_Speed_desired"
kind="output">
        <ssc:Real/>
        <ssd:ConnectorGeometry x="1.0" y="0.25"/>
    </ssd:Connector>
    <ssd:Connector name="$UI_Vehicle_Acceleration_desired"
kind="output">
        <ssc:Real/>
        <ssd:ConnectorGeometry x="1.0" y="0.125"/>
    </ssd:Connector>
    <ssd:Connector name="ActualSlip[0]" kind="output">
        <ssc:Real/>
        <ssd:ConnectorGeometry x="1.0" y="0.0"/>
    </ssd:Connector>
    <ssd:Connector name="ActualSlip[1]" kind="output">
        <ssc:Real/>
        <ssd:ConnectorGeometry x="1.0" y="-0.125"/>
    </ssd:Connector>
    <ssd:Connector name="ActualSlip[2]" kind="output">
        <ssc:Real/>
        <ssd:ConnectorGeometry x="1.0" y="-0.25"/>
    </ssd:Connector>
    <ssd:Connector name="ActualSlip[3]" kind="output">
        <ssc:Real/>
        <ssd:ConnectorGeometry x="1.0" y="-0.375"/>
    </ssd:Connector>
    <ssd:Connector name="DesiredSlipActualSlip[0]"
kind="output">
        <ssc:Real/>
        <ssd:ConnectorGeometry x="1.0" y="-0.5"/>
    </ssd:Connector>
    <ssd:Connector name="DesiredSlipActualSlip[1]"
kind="output">
        <ssc:Real/>
        <ssd:ConnectorGeometry x="1.0" y="-0.625"/>
    </ssd:Connector>
    <ssd:Connector name="DesiredSlipActualSlip[2]"
kind="output">
        <ssc:Real/>
        <ssd:ConnectorGeometry x="1.0" y="-0.75"/>
    </ssd:Connector>
    <ssd:Connector name="DesiredSlipActualSlip[3]"
kind="output">
        <ssc:Real/>
        <ssd:ConnectorGeometry x="1.0" y="-0.875"/>
    </ssd:Connector>
    <ssd:Connector name="DesiredSlip_" kind="output">
        <ssc:Real/>
        <ssd:ConnectorGeometry x="1.0" y="-1.0"/>
    </ssd:Connector>
</ssd:Connectors>
<ssd:ElementGeometry x1="-101.57274949550629" y1="-
39.110637187957764" x2="8.329537305203791" y2="70.0179074138082"/>
<ssd:ParameterBindings>
    <ssd:ParameterBinding type="application/x-ssp-parameter-
set" source="resources/PID_Controller_Simulink_FMU.ssv"/>

```

```

        </ssd:ParameterBindings>
    </ssd:Component>
</ssd:Elements>
<ssd:Connections>
    <ssd:Connection startElement="PID_Controller_Simulink_FMU"
startConnector="$UI_Controller_1" endElement="Simpack_MBS_FMU"
endConnector="$UI_Controller_1">
        <ssd:ConnectionGeometry pointsX="58.97778868675232
58.97778868675232" pointsY="56.37683987617493 56.25126600265503"/>
    </ssd:Connection>
    <ssd:Connection startElement="PID_Controller_Simulink_FMU"
startConnector="$UI_Controller_2" endElement="Simpack_MBS_FMU"
endConnector="$UI_Controller_2">
        <ssd:ConnectionGeometry pointsX="58.97778868675232
58.97778868675232" pointsY="42.735772132873535
42.610180377960205"/>
    </ssd:Connection>
    <ssd:Connection startElement="PID_Controller_Simulink_FMU"
startConnector="$UI_Controller_3" endElement="Simpack_MBS_FMU"
endConnector="$UI_Controller_3">
        <ssd:ConnectionGeometry pointsX="58.97778868675232
58.97778868675232" pointsY="29.0947026014328 28.969092965126038"/>
    </ssd:Connection>
    <ssd:Connection startElement="PID_Controller_Simulink_FMU"
startConnector="$UI_Controller_4" endElement="Simpack_MBS_FMU"
endConnector="$UI_Controller_4">
        <ssd:ConnectionGeometry pointsX="58.97778868675232
58.97778868675232" pointsY="15.453634858131409 15.32800555229187"/>
    </ssd:Connection>
    <ssd:Connection startElement="PID_Controller_Simulink_FMU"
startConnector="$UI_Vehicle_Position_desired"
endElement="Simpack_MBS_FMU"
endConnector="$UI_Vehicle_Position_desired">
        <ssd:ConnectionGeometry pointsX="58.97778868675232
58.97778868675232" pointsY="1.8125670030713081
1.6869190335273743"/>
    </ssd:Connection>
    <ssd:Connection startElement="PID_Controller_Simulink_FMU"
startConnector="$UI_Vehicle_Speed_desired"
endElement="Simpack_MBS_FMU"
endConnector="$UI_Vehicle_Speed_desired">
        <ssd:ConnectionGeometry pointsX="58.97778868675232
58.97778868675232" pointsY="-11.828500628471375 -
11.954167485237122"/>
    </ssd:Connection>
    <ssd:Connection startElement="PID_Controller_Simulink_FMU"
startConnector="$UI_Vehicle_Acceleration_desired"
endElement="Simpack_MBS_FMU"
endConnector="$UI_Vehicle_Acceleration_desired">
        <ssd:ConnectionGeometry pointsX="58.97778868675232
58.97778868675232" pointsY="-25.469568371772766 -
25.59525489807129"/>
    </ssd:Connection>
    <ssd:Connection startElement="Simpack_MBS_FMU"
startConnector="$Y_VehicleSpeed"
endElement="PID_Controller_Simulink_FMU"
endConnector="$Y_VehicleSpeed">
        <ssd:ConnectionGeometry pointsX="254.90967750549316
254.90967750549316 -142.24968910217285 -142.24968910217285"

```

```

254.90967750549316 -142.24968910217285 -142.24968910217285"
pointsY="5.701408088207245 106.95777654647827 106.95777654647827
51.82981610298157"/>
  </ssd:Connection>
  <ssd:Connection startElement="Simpack_MBS_FMU"
startConnector="$Y_w_BG02_RS2"
endElement="PID_Controller_Simulink_FMU"
endConnector="$Y_w_BG02_RS2">
  <ssd:ConnectionGeometry pointsX="245.21512985229492
245.21512985229492 -129.53850746154785 -129.53850746154785"
pointsY="-32.81700611114502 -61.65132522583008 -61.65132522583008 -
20.92254638671875"/>
  </ssd:Connection>
  <ssd:Connection startElement="Simpack_MBS_FMU"
startConnector="$Y_w_BG02_RS1"
endElement="PID_Controller_Simulink_FMU"
endConnector="$Y_w_BG02_RS1">
  <ssd:ConnectionGeometry pointsX="252.039155960083
252.039155960083 -140.16222953796387 -140.16221523284912 -
140.16221523284912" pointsY="-26.397671699523926 -72.54021406173706
-72.54021406173706 -72.51807689666748 -2.7344512939453125"/>
  </ssd:Connection>
  <ssd:Connection startElement="Simpack_MBS_FMU"
startConnector="$Y_w_BG01_RS2"
endElement="PID_Controller_Simulink_FMU"
endConnector="$Y_w_BG01_RS2">
  <ssd:ConnectionGeometry pointsX="261.0951519012451
261.0951519012451 261.0951519012451 -155.79774856567383 -
155.79776287078857 -155.79776287078857" pointsY="-19.97833549976349
-20.046536922454834 -81.87875747680664 -81.87875747680664 -
81.86542510986328 15.453636646270752"/>
  </ssd:Connection>
  <ssd:Connection startElement="Simpack_MBS_FMU"
startConnector="$Y_w_BG01_RS1"
endElement="PID_Controller_Simulink_FMU"
endConnector="$Y_w_BG01_RS1">
  <ssd:ConnectionGeometry pointsX="272.79759407043457
272.79759407043457 -173.97013664245605 -173.9701223373413"
pointsY="-13.559001088142395 -92.35836982727051 -92.35836982727051
33.6417281627655"/>
  </ssd:Connection>
</ssd:Connections>
  <ssd:SystemGeometry x1="-119.03093646036947" y1="-
51.41769753959017" x2="236.212916063884" y2="82.19926363218377"/>
</ssd:System>
  <ssd:DefaultExperiment startTime="0.0" stopTime="5.9"/>
</ssd:SystemStructureDescription>

```

Appendix 3: An example XML-based slave description file belonging to the DCP standard.

```

<?xml version="1.0" encoding="UTF-8"?>
<dcpslaveDescription dcpMajorVersion="1" dcpMinorVersion="0"
dcpSlaveName="simulink_pid" uuid="0d7217ea-ac72-11ea-bb37-
0242ac130002" variableNamingConvention="flat" >
  <OpMode>
    <SoftRealTime/>
    <NonRealTime />
  </OpMode>

  <TimeRes>
    <Resolution numerator="10" denominator="10000"
fixed="false" />
    <Resolution numerator="10" denominator="100"
fixed="false" />
    <Resolution numerator="5" denominator="100"
fixed="false" />
  </TimeRes>

  <TransportProtocols>
    <UDP_IPv4 maxPduSize="65507" >
      <Control host="128.131.136.1" port="11111"
/>
      <DAT_input_output host="" >
        <AvailablePortRange from="2048"
to="65535" />
      </DAT_input_output>
      <DAT_parameter host="" >
        <AvailablePortRange from="2048"
to="65535" />
      </DAT_parameter>
    </UDP_IPv4>
  </TransportProtocols>
  <CapabilityFlags canAcceptConfigPdus="true"
canHandleReset="true" canHandleVariableSteps="true"
canProvideLogOnRequest="true" canProvideLogOnNotification="true" />
  <Variables>
    <Variable name="_UI_Controller_1"
valueReference="1" >
      <Output>
        <Float64 start="0.0" />
      </Output>
    </Variable>
    <Variable name="_UI_Controller_2"
valueReference="2" >
      <Output>
        <Float64 start="0.0" />
      </Output>
    </Variable>
    <Variable name="_UI_Controller_3"
valueReference="3" >
      <Output>
        <Float64 start="0.0" />
      </Output>
    </Variable>
    <Variable name="_UI_Controller_4"

```

```

valueReference="4" >
    <Output>
        <Float64 start="0.0" />
    </Output>
</Variable>
<Variable name="_UI_Vehicle_Acceleration_desired"
valueReference="5" >
    <Output>
        <Float64 start="0.0" />
    </Output>
</Variable>
<Variable name="_UI_Vehicle_Position_desired"
valueReference="6" >
    <Output>
        <Float64 start="0.0" />
    </Output>
</Variable>
<Variable name="_UI_Vehicle_Speed_desired"
valueReference="7" >
    <Output>
        <Float64 start="0.0" />
    </Output>
</Variable>
<Variable name="_Y_VehicleSpeed" valueReference="8"
>
    <Input>
        <Float64 start="0.0" />
    </Input>
</Variable>
<Variable name="_Y_w_BG01_RS1" valueReference="9" >
    <Input>
        <Float64 start="0.0" />
    </Input>
</Variable>
<Variable name="_Y_w_BG01_RS2" valueReference="10" >
    <Input>
        <Float64 start="0.0" />
    </Input>
</Variable>
<Variable name="_Y_w_BG02_RS1" valueReference="11" >
    <Input>
        <Float64 start="0.0" />
    </Input>
</Variable>
<Variable name="_Y_w_BG02_RS2" valueReference="12" >
    <Input>
        <Float64 start="0.0" />
    </Input>
</Variable>
<Variable name="Parameters.DesiredSlip_Value"
valueReference="13" variability="tunable" >
    <Parameter>
        <Float64 start="0.02" />
    </Parameter>
</Variable>
<Variable name="Parameters.InitialSlip_Value"
valueReference="14" variability="tunable" >
    <Parameter>
        <Float64 start="0.0" />
    </Parameter>

```

```

        <Parameter>
            <Float64 start="0.0" />
        </Parameter>
    </Variable>
    <Variable name="Parameters.WheelRadiusm_Value"
valueReference="15" variability="tunable" >
        <Parameter>
            <Float64 start="0.46" />
        </Parameter>
    </Variable>
    <Variable name="Parameters.P_Parameter_Value"
valueReference="16" variability="tunable" >
        <Parameter>
            <Float64 start="20000.0" />
        </Parameter>
    </Variable>
    <Variable name="Parameters.D_Parameter_Value"
valueReference="17" variability="tunable" >
        <Parameter>
            <Float64 start="500.0" />
        </Parameter>
    </Variable>
    <Variable name="Parameters.N_Parameter_Value"
valueReference="18" variability="tunable" >
        <Parameter>
            <Float64 start="100.0" />
        </Parameter>
    </Variable>
    <Variable name="Parameters.I_Parameter_Value"
valueReference="19" variability="tunable" >
        <Parameter>
            <Float64 start="700000.0" />
        </Parameter>
    </Variable>
    <Variable name="Parameters.IntegrationtoPosition_IC"
valueReference="20" variability="tunable" >
        <Parameter>
            <Float64 start="0.0" />
        </Parameter>
    </Variable>
    <Variable name="Parameters.ConstantVehicleAccelerationms2_"
valueReference="21" variability="tunable" >
        <Parameter>
            <Float64 start="1.0" />
        </Parameter>
    </Variable>
    <Variable name="Parameters.InitialVehicleSpeedms_Value"
valueReference="22" variability="tunable" >
        <Parameter>
            <Float64 start="18.0" />
        </Parameter>
    </Variable>
</Variables>
<Log>

    <Categories>
        <Category id="1" name="DCP_SLAVE_PID" />
    </Categories>
<Templates>
    <Template id="1" category="1" level="3"

```

```

msg="[Time = %float64]: sin(%uint64 + %float64) = %float64" />
    </Templates>
</Log>
</dcpSlaveDescription>

```

Appendix 4: An example OSP system structure description XML file.

```

<?xml version="1.0" encoding="utf-8" ?>
<OspSystemStructure

xmlns="http://opensimulationplatform.com/MSMI/OSPSystemStructure"
    version="0.1">
    <BaseStepSize>0.001</BaseStepSize>
    <Simulators>
        <Simulator name="MBSVehicle"
            source="proxyfmu://localhost?file=../MBSVehicle.fmu"/>
        <Simulator name="PIDController"
            source="proxyfmu://localhost?file=../PIDController.fmu"/>
    </Simulators>
    <Connections>
        <VariableConnection>
            <Variable simulator="MBSVehicle" name="$Y_VehicleSpeed"/>
            <Variable simulator="PIDController" name="$Y_VehicleSpeed"/>
        </VariableConnection>
        <VariableConnection>
            <Variable simulator="MBSVehicle" name="$Y_w_BG01_RS1"/>
            <Variable simulator="PIDController" name="$Y_w_BG01_RS1"/>
        </VariableConnection>
        <VariableConnection>
            <Variable simulator="MBSVehicle" name="$Y_w_BG01_RS2"/>
            <Variable simulator="PIDController" name="$Y_w_BG01_RS2"/>
        </VariableConnection>
        <VariableConnection>
            <Variable simulator="MBSVehicle" name="$Y_w_BG02_RS1"/>
            <Variable simulator="PIDController" name="$Y_w_BG02_RS1"/>
        </VariableConnection>
        <VariableConnection>
            <Variable simulator="MBSVehicle" name="$Y_w_BG02_RS2"/>
            <Variable simulator="PIDController" name="$Y_w_BG02_RS2"/>
        </VariableConnection>
        <VariableConnection>
            <Variable simulator="MBSVehicle" name="$UI_Controller_1"/>
            <Variable simulator="PIDController" name="$UI_Controller_1"/>
        </VariableConnection>
    </Connections>
</OspSystemStructure>

```

```

</VariableConnection>
<VariableConnection>
  <Variable simulator="MBSVehicle" name="$UI_Controller_2"/>
  <Variable simulator="PIDController" name="$UI_Controller_2"/>
</VariableConnection>
<VariableConnection>
  <Variable simulator="MBSVehicle" name="$UI_Controller_3"/>
  <Variable simulator="PIDController" name="$UI_Controller_3"/>
</VariableConnection>
<VariableConnection>
  <Variable simulator="MBSVehicle" name="$UI_Controller_4"/>
  <Variable simulator="PIDController" name="$UI_Controller_4"/>
</VariableConnection>
<VariableConnection>
  <Variable simulator="MBSVehicle"
name="$UI_Vehicle_Position_desired"/>
  <Variable simulator="PIDController"
name="$UI_Vehicle_Position_desired"/>
</VariableConnection>
<VariableConnection>
  <Variable simulator="MBSVehicle"
name="$UI_Vehicle_Speed_desired"/>
  <Variable simulator="PIDController"
name="$UI_Vehicle_Speed_desired"/>
</VariableConnection>
<VariableConnection>
  <Variable simulator="MBSVehicle"
name="$UI_Vehicle_Acceleration_desired"/>
  <Variable simulator="PIDController"
name="$UI_Vehicle_Acceleration_desired"/>
</VariableConnection>
</Connections>
</OspSystemStructure>

```

Appendix 5: An example JSON file based on the R4F standard.

```
{
  "uc-parameters": [
    {
      "id": "$G_Scenario_Params.$_Track_Bogenradius",
      "value": 10000.0
    },
    {
      "id": "$G_Scenario_Params.$_Track_Superelevation",
      "value": 0.0
    },
    {
      "id": "$G_Scenario_Params.$_Vehicle_Speed",
      "value": 120.0
    },
    {
      "id": "$G_Scenario_Params.$_Masse_Passagiere",
      "value": 15000.0
    },
    {
      "id": "$G_Scenario_Params.$_Rail_Wheel_Fric_Coeff",
      "value": 0.1
    }
  ],
  "uc-metadata": [
    {
      "id": "Multibody Simulation (MBS) of a Railway Vehicle",
      "description": "...",
      "filetype": "INPUT",
      "input_parameters": [
        {
          "id": "$G_Scenario_Params.$_Track_Bogenradius",
          "name": "$G_Scenario_Params.$_Track_Bogenradius",
          "type": "float",
          "unit": "m",
          "info": [
            {
              "dependency": ""
            }
          ]
        },
        {
          "id": "$G_Scenario_Params.$_Track_Superelevation",
```


A.2 Technical Documentation

A.2.1 Functional Mock-up Interface

The FMU container contains an XML model description file, binary files for Operating System (OS) dependency, and optional source codes [38]. The FMU either may have its own simulation solver (FMI for Co-Simulation) or needs to be started in a simulation environment (FMI for Model Exchange) [38]. The FMU can be generated due to the default software tool used for the model simulation, and then executed by some particular software tools, which are also listed in the Tools section of the FMI Homepage [41, 133].

Appendix 6: General structure of the ZIP-formatted FMU file. [38]

File / Folder	Description
modelDescription.xml	Description of FMU (required)
model.png	Optional image file of FMU icon
documentation index.html	Optional directory containing the FMU documentation Entry point of the documentation
sources	Optional directory containing all C resources all needed C sources and C header files to compile and link the FMU with exception of: fmi2TypesPlatform.h, fmi2FunctionTypes.h and fmi2Functions.h The files to be compiled (but not the files included from these files) have to be reported in the xml-file under the structure <ModelExchange><SourceFiles> ... and <CoSimulation><SourceFiles>

binaries Windows: win32, win64; Linux: linux32, linux64; Mac: darwin32, darwin64	Optional directory containing the binaries Optional binaries for 32-bit and 64-bit Windows, Linux & Mac
resources	Optional resources needed by the FMU data in FMU specific files which will be read during initialization; also more folders can be added under resources (tool/model specific). In order for the FMU to access these resource files, the resource directory must be available in unzipped form and the absolute path to this directory must be reported via argument "fmuResourceLocation" via fmi2Instantiate.

In Appendix 6, different files and folders are listed and described, which build the general structure of the zip file of an FMU. First, the modelDescription.xml file gives an overview of all the metadata required to build the FMU (for an example see Appendix 1). Second, the visual picture of the model packaged into FMU can be optionally shown with the model.png file. Third, documentation folder incl. the index.html file can be optionally used to keep the FMU documentation in the FMU container. Besides, the optional sources and binaries folders are used for the distribution of the FMU, at least one of which must exist as one implementation. The binaries folder contains binary files, which make the FMU distribution suitable for a particular machine based on its OS (e.g. Windows x64), which is essential for running a target simulator with available licenses of run-time libraries as an example. The sources folder has all C resources needed for FMU compilation and linking, which is important to translate and download for a particular micro-processor as an example. Lastly, the resources folder includes additional data provided in different formats (e.g. table, maps) and read into the model, which was packed into the FMU, during initialization. [38]

Appendix 7: The entire content of the modelDescription.xml file in the FMU.

```

<fmiModelDescription xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
modelName="Drivetrain" guid="{e776f230-2966-4f34-a70d-20782ef54542}" ge
(64-bit), 2
numberOfEve
00:30Z" vari
  <ModelExc
    providesDirectionalDerivative="true"
    <SourceFiles>
      <File name="all.c"/>
    </SourceFiles>
  </ModelExc>
  <CoSimula
    canInterp
    canSerial
    canGetAndSet
    MUstate="true"
  </CoSimulation>
  <UnitDefi
    <Unit n
      <Base
    </Unit>
    <Unit name="N.m.s/rad">
      <BaseUnit kg="1" m="2" s="-1" rad="-1"/>
    </Unit>
    <Unit name="N.m.s/rad">
      <Base
    </Unit>
    <Unit n
      <BaseUnit kg="1" m="2"/>
    </Unit>
  </UnitDefi
  </fmiModelDescription>
  
```

In Appendix 7, the typical XML structure of a model description file is shown. The model description file includes model metadata (e.g., model name, software tool, FMI version number, ...), input parameters, input and output channels. All these ports are previously assigned to the model for result extraction (for result analysis and validation purposes) and latter co-simulation with another simulation model through connections, where the ports are connected to the ports of the other model, in some cases. The input parameters are previously created in the raw model as well in order users to parameterize the FMU model in their own virtual platform.

A.2.2 System Structure and Parameterization

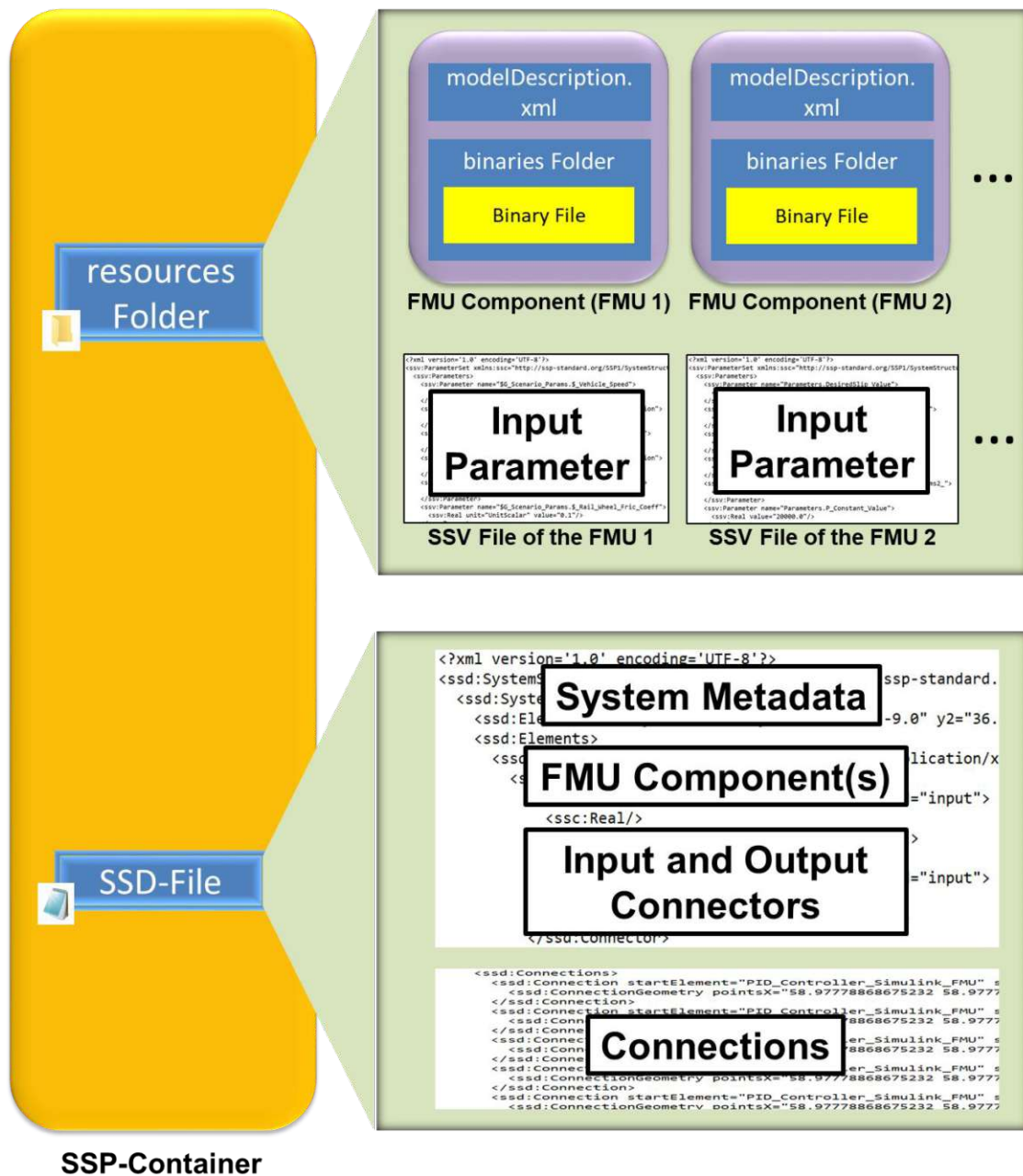
Appendix 8: General structure of the ZIP-formatted SSP file. [49]

File / Folder	Description
SystemStructure.ssd	Description of system (required)
documentation	Optional directory containing the SSP documentation
index.html	Entry point of the documentation

extra	Optional extra folder Including additional entries for more system control
resources FMU file (*.fmu), SSV file (*.ssv), SSM file (*.ssm), SSB file (*.ssb)	Optional resources folder Including FMU file(s), parameter set(s), parameter mapping(s), signal directory(ies)

In Appendix 8, all the files and folders building the general structure of an SSP container are listed with corresponding descriptions in details. First, there is a SystemStructure.ssd file, which is a System Structure Description (SSD) file, required and identifies the entire system of the co-simulation model by defining input & output channels of FMU component(s), and connections between them (for an example see Appendix 2). Second, the documentation folder can optionally be used to store the SSP documentation with the index.html file, which is employed as an entry point and must exist if the documentation folder is provided. Then, the extra folder may be used to provide additional entries incl. additional data and metadata, which help to control the entire system. Lastly, the optional resources folder includes the FMU component(s) and may contain optional System Structure Parameter Values (SSV), System Structure Parameter Mappings (SSM) or System Structure Signal Dictionaries (SSB) files. The SSV file includes the entire parameter set with parameter values, which are directly applied to the corresponding model as default. Surely, other methods can be figured out to do the input parameterization like using an input JSON file, that was succeeded. The SSM file has a parameter mapping, which helps to map the parameter names in a parameter source to the parameter names in the system or component. This can be optionally used to transform the value of these parameters before these are read to the system or component. The SSB file contains a signal directory, which is a collection of signals handled as a signal bus (like a CAN-bus in embedded systems) and used to define signal connections. [128]

Appendix 9: Main components and their content of the SSP file incl. the FMU component(s).



Appendix 9 shows and describes the main components of a typical SSP file, which is used for the asset integration task mentioned in [Subsection 4.1.1](#). First, there are the FMUs and their belonging SSV files describing the input parameters. Then, the typical SSD file includes system metadata such as XML version, SSP version, encoding format (UTF-8), system name, SSP generation tool, generation date and time, and URIs of the classes belonging to the SSP.

A.2.3 Distributed Co-Simulation Protocol

In Appendix 10, an overview of the general components, used for the XML-based slave description file, are shown, which are necessary to implement a DCP slave into a co-simulation scenario (for an example see Appendix 3). Krammer and Benedikt [74] proposed to use this slave description file to describe the co-simulation scenario and to configure the slave based on the DCP standard. These components are described below (for more information see [25]):

1- XML Attributes: Meta data identifying and characterizing the DCP slave description element (e.g., dcpSlaveName, author, generation tool).

2- Operating Modes (OpMode): Types of valid operations of the DCP slave, which are:

- HardRealTime,
- SoftRealTime,
- NonRealTime.

3- Unit Definitions: A global list of units used in the DCP slave (e.g., meter (m)).

4- Type Definitions: A global list of element types used in the DCP slave (e.g., Int16, Uint32).

5- Vendor Annotations: Optional information, which is relevant to vendor. (e.g., tool name)

6- Time Resolutions (TimeRes): Time resolutions and resolution ranges, which are authorized for the DCP slave.

7- Heartbeat: Capability of the DCP slave to monitor heartbeat signals coming from the DCP master (optional).

8- Transport Protocols: Communication protocols used for the DCP slave as DCP drivers. These are:

- UDP_IPv4,
- CAN,
- USB2,
- Bluetooth,
- TCP_IPv4.

9- Capability Flags: Abilities of the DCP slave (e.g., canMonitorHeartbeat).

10- Variables: All data of the DCP slave, which are open to access via DCP. These are:

- Input,
- Output,
- Parameter,
- StructuralParameter,
- Annotations.

11- Logs: Categories and templates of the DCP slave used for logging.

Appendix 10: The entire content of an XML-based slaveDescription.dcp file. [25]

The diagram illustrates the structure of an XML-based slaveDescription.dcp file. The XML content is shown in the background, with various components highlighted in boxes:

- XML Attributes**: Located at the top of the XML document.
- Operating Modes**: A section containing mode definitions.
- Unit Definitions**: A section containing unit definitions.
- Type Definitions**: A section containing type definitions.
- Vendor Annotations**: A section containing vendor-specific annotations.
- Time Resolutions**: A section containing time resolution definitions.
- Transport Protocols**: A section containing transport protocol definitions.
- Heartbeat**: A section containing heartbeat definitions.
- Capability Flags**: A section containing capability flags.
- Variables**: A section containing variable definitions.
- Logs**: A section containing log definitions.

The XML content is as follows:

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <dcpSlaveDescription xmlns="http://www.ti.com/xml/dcpSlaveDescription" slaveName="Slave"
3   variableName="Slave"
4   <OpMode>
5   <SoftRealTime>
6   </OpMode>
...
15 <UDP_IPv4 maxPduSize="65507" >
...
188 <Parameter>
189   <Float32 start="3 " />
190 </Parameter>
191 </Variable>
192 </Variables>
193 <Log>
194 <Categories>
195   <Category id="1" name="DCP_SLAVE" />
196 </Categories>
197 </Log>
198 </Log>
199 </Log>
200 </Log>
201 </Log>
202 </Log>
203 </Log>
204 </Log>
205 </Log>
206 </dcpSlaveDescription>

```

A.2.4 Open Simulation Platform

The OSP software consists of the main components:

1- OSP *libcosim*: A C/C++ co-simulation library, which can readily be applied to any simulation software to run co-simulations there. [51]

2- Demo applications and examples: There are tools and use case examples (e.g., simulation of a marine vessel operating a crane) to demonstrate co-simulations through the use of the OSP *libcosim* library. [48]

3- OSP-IS: An additional OSP property, providing an ontology-based interface between the FMU and OSP to ease and accelerate the co-simulation process by adding semantics.

4- OSP reference models: A set of representative models belonging to the maritime equipment (e.g., crane model, winch model,...). These are applied as example FMUs and simulation system configurations to test demo applications with *libcosim* and OSP-IS implementations. [47, 81]

To build the co-simulation environment, key software tools are used, designed and developed to co-simulate different simulation models, or even to execute these separately (single simulation) in a distributed fashion. Some of these tools are: [97]

- *libcosim*: A C++ library, providing integration of co-simulation into any software as previously mentioned. [51]
- *proxy-fmu*: A tool-independent solution, written in C++ and helping to execute FMUs in a distributed fashion and auto-spawn new processes from the FMU instantiations in other machines. [54]
- *cosim-cli*: A co-simulation tool, used in the CLI after installing and building it. This tool helps to run co-simulations from the CLI or other applications. Besides, FMUs can be tested and debugged by using it. [46]
- *cosim-demo-app*: A demo application of the *libcosim* C++ library. This application is based on server-client, where the server is written in Go, and the client in clojurescript. [47]
- *libcosimc*: A C library, providing an OSP co-simulation API by using functions of the *libcosim* C++ library. [52]
- *osp-validator*: An ontology-based Java application, helping to validate the FMUs and simulation configurations, which are described in the co-simulation system structure regarding to OSP-IS. [53]

The structure of the co-simulation system is described either via the SSP standard, where the SSD file is applied (see [Appendix Subsection A.2.2](#)) and is suitable for co-simulations of multiple FMUs, or OSP system structure format (for an example see Appendix 4), which uses the OSP-IS and is suitable for single simulations (single FMU). [97]

A.2.5 The R4F Ontology

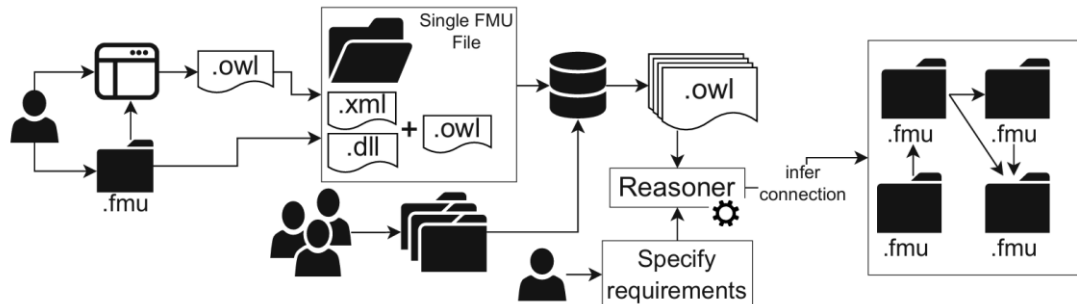
Kern [72] designed and developed an ontology-based knowledge model of the R4F domain, which is a railway domain considered for the Rail4Future project and includes railway vehicle and track. For the implementation of the R4F Ontology in the platform, he used the Ontology Web Language (OWL) standard to enable the reasoning ability, based on first-order logic, in the ontology. He defined and described steps based on the NeOn methodology [130], which he followed to develop the ontology for the project. These are:

- 1) Initiation phase,
- 2) Reuse and re-engineering phase,
- 3) Design phase,
- 4) Implementation phase.

First, in the initiation phase, he defined aim, scope, implementation language, users, usage areas and requirements (functional and non-functional). In the second phase, namely reuse and re-engineering phase, he adapted the reuse and re-engineering processes for his work, which assisted him to elevate the quality of the ontology by reusing some knowledge (e.g., time, space) and re-engineering some resources. As a concrete re-engineering example, he re-engineered the FMI specification by converting the XML-based model description file into the OWL format in order to apply the FMI standard to the R4F Ontology. In the third phase, he conceptualized the R4F Ontology by building a glossary of terms, concept classification tree, concept dictionary, binary-relation table and instance attribute table. These include terms, descriptions, classes, sub-classes, attributes, concepts and relations between these concepts, which he successfully defined from the R4F domain and then adapted to the R4F Ontology for the Rail4Future project in general. In the last phase, he implemented the ontology into the OWL language, which is the formal language of knowledge representation, can be computed for knowledge inference and therefore allows to perform auto-reasoning based on first-order logic.

Appendix 11 indicates an overview of the workflow, which Kern [72] proposed to annotate an FMU file, to convert it into the OWL format, and then to generate a simulation topology from the OWL file through the auto-reasoning process. In this workflow, all the FMUs are successfully inferred to each other helping to extract relevant information from the R4F domain using the R4F Ontology.

Appendix 11: The workflow followed to extract information from an FMU file. [72]



Appendix 12 shows the GUI windows of the Python application, which is based on the R4F Ontology and developed by Kern [72]. In the general information window (see Fig. Appendix 13a), the users of the R4F Platform enter input data such as simulation name and author, which are required fields, to feed the ontology with those information belonging to a simulation model. They can also put file name, author email, description, maintainer, creation date and version number in this window. Then, they are able to annotate input and output variables, belonging to the model, by giving variable name, unit, quantity, semantic type, variability and description in these input variables and output variables windows (see Appendix 12b and 12c). Besides, they can select the variable in a drop-down menu of these windows to directly reach the information about the model.

Appendix 12: An ontology-based software application with GUI. [72]

Ontology Annotation UI

Menu

General Input Output

* required Fields. Hover for fieldname more infos.

Open File Name:

Simulation Name *:

Author*:

Author Email:

Description:

Maintainer:

Date Created:

Version:

Save Metadata

(a) General Information

Ontology Annotation UI

Menu

General Input Output

* required Fields. Hover for fieldname more infos.

Select Input Variable

Name*:

Unit*:

Quantity*:

Semantic Type*:

Variability:

Description:

Save Metadata

(b) Input Variables

Ontology Annotation UI

Menu

General Input Output

* required Fields. Hover for fieldname more infos.

Select Output Variable

Name*:

Unit*:

Quantity*:

Semantic Type*:

Variability:

Description:

Save Metadata

(c) Output Variables.

A.2.6 DevOps and CI/CD

Code Analysis

Six kinds of classifications of the source code analysis process are listed as it follows: [16]

1) Static vs. dynamic:

- Static code analysis:** Analyzing the source code in a static way, which means that the entire structure and components of the code are observed without considering its execution resp. dynamic

behavior. In this case, the inputs given to the code are not taken into account for this kind of code analysis.

- **Dynamic code analysis:** Analyzing the source code by executing it in a virtual environment to find out the effects impacting on the dynamical behavior of the code. This helps to predictively determine approximate outputs resulting from specific inputs entered into the code script.

2) Sound vs. unsound:

- **Sound code analysis:** Analyzing the source code by checking if the code provides valid arguments, which leads to gain valid outputs from the code execution. This assures the correctness of the code script.
- **Unsound code analysis:** Rather than the sound code analysis, the arguments of the code, and its belonging results don't need to be 100%-correct, which can be nearly or close enough to valid. It is not expected to achieve a huge number of false positives (e.g., redundancies like a bunch of comments, which are not related to concerns) from the unsound code analysis, increasing the efficiency of the correctness, compared to the sound one.

3) Safe vs. unsafe: The code analysis approach can be defined as “safe”, if the analysis provides exact responses, that are evaluated as one-sided (e.g., an assignment can't be considered for a specific use, in this case it is labeled with “not considered”).

4) Flow-sensitive vs. flow-insensitive:

- **Flow-sensitive code analysis:** Meaning that the program's flow belonging to the code can only be observed in one order. For example, a code provides the sequence $a \rightarrow b \rightarrow c$, where a can point to b , but not to c .
- **Flow-insensitive code analysis:** The insensitivity of the code analysis offers flexibility in the program's flow, which can be applied for any order. For instance, both the cases $a \rightarrow b$ and $a \rightarrow c$ might be possible, if the code analysis example, mentioned above, is flow-insensitive.

5) Context-sensitive vs. context-insensitive:

- **Context-sensitive code analysis:** Analyzing the source code in a respective manner, which means that the results of a code analysis process are redirected only to its own call-site, from where the process for the code analysis is invoked.

- **Context-insensitive code analysis:** In contrast to the context-sensitive one, the outputs of the code analysis process are distributed to all the call-sites, which are related to the process.

6) Complexity: In a source code analysis, many problems come into being and several solutions are found out to solve the problems. This causes high complexity in the code analysis regarding to the trade-off between the precision and computation effort. For example, computation of imprecise points instead of precise ones succeeds in a relatively short time while context- and flow-sensitive points cause relatively high computation time.