An FMI- and SSP-based Model Integration Methodology for a Digital Twin Platform of a Holistic Railway Infrastructure System

Ozan Kugu[1] Shiyang Zhou[1] Rebecca Nowak[2] Gabor Müller[3] Stefan H. Reiterer[3] Alexander Meierhofer[3] Stefan Lachinger[4] Lukas Wurth[1] Manfred Grafinger[1]

[1]Institute of Engineering Design and Product Development, TU Wien, 1060 Vienna, Austria, ozan.kugu@tuwien.ac.at

[2]VRVis Zentrum für Virtual Reality und Visualisierung Forschungs-GmbH, 1220 Vienna, Austria, rnowak@vrvis.at

[3]Virtual Vehicle Research GmbH, 8010 Graz, Austria, alexander.meierhofer@v2c2.at

[4]AIT - Austrian Institute of Technology GmbH, 1210 Vienna, Austria, stefan.lachinger@ait.ac.at

Abstract

Nowadays, the digitalization of large-scale railway infrastructure systems is a major trend, which helps to reduce the life-cycle costs of the railway transportation. For this purpose, the Digital Twin (DT) technology can be used to interoperate different digital data and models, belonging to the railway infrastructure system, in a virtual platform for predictive maintenance, diagnostics and condition monitoring in the railway sector. However, the simulation models of the infrastructure system are tool-dependent, lack ease-of-use and platform compatibility. Therefore, we have to customise them in order to make them more representative and then integrate easily and tool-independently into the DT platform. For this purpose, we propose to use the Functional Mock-up Interface (FMI) and System Structure Parameterization (SSP) technologies as open interface standards between the models and software tools. In this work, we demonstrate the application of the FMI and SSP standards separately for two use cases, which include a multibody simulation (MBS) model of a railway vehicle and residual life time (RLT) calculation of a steel bridge.

Keywords: FMI, FMU, SSP, Model Integration, Digital Twin, Railway Digitalization

1 Introduction

In recent decades, the railway sector has made a significant contribution to both local and long-distance public and freight transport. The holistic, large-scale railway infrastructure system has played a major role in this. The system is very complex and consists of different subsystems such as railway vehicle, track, turnout, tunnel and bridge. These are to be controlled, maintained, monitored, diagnosed and visualized, which is complex and time-consuming. Therefore, the whole system has to be easy to control and simply presented to the train operators and infrastructure managers for their operational, supervision and maintenance tasks while expanding life cycle time and reducing costs at the same time. This is certainly possible if the system can be used in a user-friendly and interactive virtual environment. Nowadays, the DT technology is foreseen as a suitable method to digitalize the railway system, as it is already applied in different kinds of systems in many sectors such as automotive, aircraft, etc., incl. the railway infrastructure. For example, (Kaewunruen, AbdelHadi, et al. 2022) proposed to use it for efficient maintenance, resilience and sustainability of railway bridge infrastructures. (Hamarat, Papaelias, and Kaewunruen 2022) also used it to analyze and simulate fatigue damage in railway turnouts. Unfortunately, there are difficulties in applying the railway infrastructure system to a DT platform by using different subsystem assets, adapters and interfaces, which is even more sophisticated when considering the proper communication and interaction of the assets with each other. As (Ahmadi et al. 2021) mentioned, uncertainties of measured system parameters and variables due to external factors (e.g. noise, weather), abnormalities caused by time-frame mismatches between system and model, and incomplete system interpretation, understanding and not 100% reliability of the DT are the overall challenges to overcome while operationally implementing the DT platform.

In order to overcome the difficulty of asset integration mentioned above, first, we propose to use the FMI standard, which helps us to adopt simulation models of different railway infrastructure subsystems, to make them easy to use, tool independent, platform compatible and suitable for intellectual property (IP) protection. FMI is an open standard, providing an interface between dynamic simulation models and various software tools, and is already supported by more than 170 tools. The models are supposed to be packed into the Functional Mock-up Unit (FMU), which is the simulation unit of the FMI consisting of a model description file (xml), implementation in source and/or binaries, additional data and functionality. (https://fmi-standard.org/)

For this work, the second standard interface we propose to use is SSP. It consists of one or more FMUs, transferable parameter description formats such as system structure definition (.SSD), signal directories (.SSB), parameter values (.SSV), parameter mapping (.SSM), all as packed in System Structure Package (.SSP) (Pierre R. Mai 2018). Many industrial and academic members also contribute to the SSP standard. The technology helps to keep and interoperate different FMUs, belonging to a physical system, together with parameter mapping, storage and exchange in a whole SSP-file. (`https://ssp-standard.org/`)

Models and data of different railway infrastructure subsystems need to interoperate with each other properly, which is a very complex and elaborate task to handle. Finally, we need to ensure user control and visual representation of the railway infrastructure under consideration of user-friendliness and user-interactivity. Therefore we will integrate these models and data into a DT platform, called R4F Platform and proposed by (Zhou, Dumss, et al. 2022).

In addition to building the FMI- and SSP-based interfaces between the models and the platform, we need to design and optimize the entire integration process of the platform, which helps us to achieve the right visual output and enhance the user control. For this work, we use Jenkins pipelines for the continuous integration (CI) of these models. Jenkins is used, because it is open-source, time-saving, user-friendly, tool-independent and provides many plugins (cf. (Mysari and Bejgam 2020)). Besides, it shows durable, pausable, versatile characteristics and its workflow is code-based (`https://www.jenkins.io/doc/book/pipeline/`). In addition, we use a pipeline auto-generation technology, which is able to automatically and dynamically generate the Jenkins pipeline from a graph data-model containing different model characteristics by using a graph database management system (DBMS), which helps us to store, version and manipulate pipeline graphs used to auto-generate CI pipelines, in the platform (see (Reiterer, Schiffer, and Benedikt 2022) for more details on the implementation). This also helps to reduce the configuration effort of the pipeline workflow by eliminating the need to write pipeline code for each change that may be applied. Most importantly, the auto-generation technology helps to maintain and manage the reusability, traceability, parameter changes and data structure applicability of different use cases, making them even more understandable to the end user.

This paper aims to show how to seamlessly integrate models into the R4F Platform by using both the FMI and SSP standards as a model integration methodology. In this work, first section 2 shows examples of related research on DT for railways and the application of the standards. In section 3, we explain the proposed model standardisation and simulation approaches to integrate the models into the platform in detail. In section 4, we describe the FMU and SSP export, simulation process and result comparison of the aforementioned two use cases as a demonstration of the two approaches. In section 5, we show the proposed integration process, which helps to realize the complete model integration in the platform. Finally, in section 6, we present the conclusion and outlook of this work.

2 Related Work

2.1 Digital Twin for Railways

Using a DT has the potential to supervise and regulate a physical system in real time, leading to enhanced system performance, decreased maintenance expenditures, and enhanced safety. In the railway industry, DT can be applied to multiple areas such as predictive maintenance, fault diagnosis, dynamic analysis, and condition monitoring. This technology incorporates data from multiple sources, including sensors, cameras, and historical records, to create a unified platform capable of modeling and simulating the physical system.

In 2018, (Kaewunruen and Xu 2018) investigated a DT-aided Building Information Modelling (BIM) application that was used to adopt a 3D model of a station building and transform it into a 6D building information model for planning, design and operational purposes. During their research, they identified some limitations and risks of the BIM adoption such as the lack of a model standard, IP issues (copyright, data privacy), relatively high project costs and lack of model accuracy due to inaccurate modeling and data entry control, high model complexity and inaccurate design data. These drawbacks need to be considered and overcome when implementing a DT application. After that (Kaewunruen and Lian 2019) established and developed a 6D BIM of a railway turnout system by using the DT technology to enhance information flow, visualized maintenance, cost estimation and collaboration among different stakeholders in terms of life cycle management, for railway turnout systems. These insights can benefit engineers, project managers, technicians, and senior management.

(S. Zhang et al. 2021) created a framework for DT-assisted fault diagnostics and real-time health monitoring of railway point machines. However they focus on a single subsystem of the railway infrastructure, from which models and data of different subsystems are to be integrated into the DT appropriately to make the DT system more comprehensive. (Zhou, Dumss, et al. 2022) also proposed to continuously integrate the models and data from different railway subsystems into the conceptual model-based R4F Platform, which represents the fully connected and digital version of the holistic railway infrastructure system with a 6-layer system architecture. This work aims to enhance the model integration process occurring in the platform as mentioned before.

2.2 Functional Mock-up Interface (FMI)

The FMI standard came up first in 2010 as a new open-source interface standard for Model Exchange (ME) 1.0 (`https://fmi-standard.org/assets/releases/FMI_for_ModelExchange_v1.0.pdf`) to export different dymamic system models by generating C-code either in source or binary form (Balakirsky et al. 2010). After that, (Bastian et al. 2011) proposed to use the

FMI for Co-Simulation (CS), consisting of co-simulation interface and description schema, to simulate coupled subsystems time-dependently. After that (Blockwitz et al. 2012) introduced the FMI 2.0 providing further features such as combination and unification of both ME and CS interfaces in one document (modelDescription.xml), enhancement of the interface variables and their classification (causality & variability, parameter tuning during simulation), saving and restoring the FMU state, improved dependency description by using an element called ModelStructure, Jacobian matrices (e.g. for implicit integration methods or FMU linearization) and improved unit definitions. In 2014, (Bertsch, Ahle, and Schulmeister 2014) evaluated the FMI technology by using the FMU Compliance Checker and FMI Cross Checking methods to further improve the maturity of FMI-based simulations. They also found the technology very promising for sharing different simulation models between different stakeholders in collaboration with OEMs (Original Equipment Manufacturers) from an industrial perspective.

(Thule et al. 2018) used the FMI standard for their work, where they implied the high potential of the technology to co-simulate different simulation models with an FMI-suitable tool. Besides, (Pieper and Obermaisser 2018) suggested applying the FMI method to the Software-in-the-Loop simulation via the Internet or LANs, which is used to test networked railway systems, to provide an interface for the simulation supported by many different tools. In this way they were able to increase the tool-independency of the simulation, which is also one of the main reasons for using the standard for this work. Furthermore, (Hartmann 2020) managed to co-simulate a couple of two subsystems (a physical barge and crane), implemented in a DT system, by using the FMI methodology. He also found the results very promising due to their high accuracy although further validation of the results in a real-life environment and the development of a real-time DT co-simulation are needed in the future. In addition, (Golightly et al. 2022) used FMI to design a rail multi-model for rail decarbonisation. They also suggested using the technology to overcome IP issues due to different software tools, models and skills, although there are some limitations such as limited modelling quality, validation of results and lack of comparison between multi-modelling and single model.

Based on the aforementioned research, the FMI methodology shows a great potential for adopting different simulation models for co-simulation in a DT platform. Therefore, this work aims to apply the technology to different models of the railway subsystems such as the railway vehicle or railway steel bridge belonging to the two use cases for their integration into the R4F Platform.

2.3 System Structure and Parameterization (SSP)

The SSP standard was first presented at the 1st Japanese Modelica Conference 2016 (`https://ssp-standard.org/literature/`) as a standardized format to connect a network of components such as FMUs, to store and apply their parameters to these components, and also to ensure protection of the IP of these parameters. (Ochel et al. 2019) developed an application called OMSimulator to parameterize, compose and exchange FMI-based models for both co-simulation and model exchange by using the SSP technology in the application. As another example, (Hällqvist et al. 2021) used the SSP standard for parameter specifications and exchange between different simulation and geometric models interoperating in an aircraft vehicle system and therefore found it very promising to develop their automated simulation method.

For this work, we also consider the SSP technology promising for the further development of the co-simulation of one or more railway subsystem FMUs with additional supported parameter description sets. Therefore, this research paper aims to demonstrate the methodology by packaging all the FMU and parameter files into one file, belonging to each of the two use cases. The one file will then be used for its FMU-based simulation as integrated into the R4F Platform.

3 Methodology

In this section, first, we describe the approach followed to design and optimize the interface between the model and R4F Platform by using both the FMI and SSP standards (see Figure 1). Secondly, we explain the path we followed to realize the simulation of the FMI- and SSP-standardized model in the platform in Figure 2, which helps to demonstrate both previously mentioned use cases for the railway digitalization in this work.

3.1 Model Standardisation Approach

On the left side of Figure 1, we specify the available models belonging to the various railway infrastructure subsystems as the first step. These models consist of default input parameters, simulation algorithm and output channels, which are connected to each other, ready to be simulated and visually demonstrated by the default software tool of the model. In the next step, they are formatted into the FMU either with solver (FMI for Co-Simulation) or without (FMI for Model Exchange), which is possible by using suitable FMI tools (`https://fmi-standard.org/tools/`). This makes FMI-standardised models more tool-independent, easier to use and more platform-compatible, because the end user doesn't need to open the model's default software tool for simulation, but can easily redirect the FMU file anywhere else and then bring it into simulation inside the R4F Platform. The FMU file consists of a required model description XML file, which describes the model (incl. the registered inputs and outputs), an optional binaries folder, including file(s) for operating system (OS) compatibility of the FMU, an optional sources folder with all C-sources used for FMU compilation and linking, and an optional resources folder with resources needed by the FMU to read data from model specific files during initialization ((https://fmi-standard.org/assets/releases/FMI_for_ModelExchange_and_CoSimulation_v2.0.pdf), p. 66). As the last step, the FMU is packed into the SSP format, which can be exported by a limited number of tools, most of which are already commercial (https://ssp-standard.org/tools/). The SSP file consists of one or more FMUs representing all the simulation models of the railway infrastructure system, which shows a way to interoperate the models with each other by using one file in the platform. The SSP also includes the SSD and SSV files, which are important to describe, easily exchange, keep input and output parameters and apply them into the FMU(s). In addition, we create an extra folder in the SSP file, where we separately upload additional necessary input data sets and simulated results. The reason for keeping these input data sets separate is to keep all the input data modular and reusable, thus reducing data complexity. This also helps the end user to easily find what they want to see and set up as input. The simulated results, which are the model simulation results from the default software tool, are also needed to confirm the reliability of the SSP simulation by comparing them with the SSP simulation results. All these aspects of the SSP mentioned above are essential for integrating the models into the R4F Platform in terms of easy and platform-independent configuration, mapped description, backup storage and IP protection of the parameters for this work.

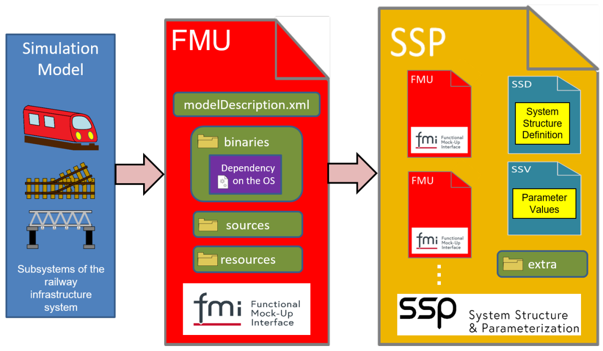

Simulation Model
Subsystems of the railway infrastructure system
FMU
modelDescription.xml
binaries
Dependency on the OS
sources
resources
fmi Functional Mock-Up Interface
SSP
FMU
SSD
System Structure Definition
SSV
Parameter Values
extra
ssp System Structure & Parameterization

Figure 1. Model standardisation approach.

3.2 Model Simulation Approach

Figure 2 shows how we implement the simulation process of the SSP-standardized model in the R4F Platform. The SSP file of the model consists of the FMU file(s), the SSV file for input parameterization, simulated results, additional input data sets and the SSD file with the preregistered output channels needed for result analysis, validation and visualization to the end user.

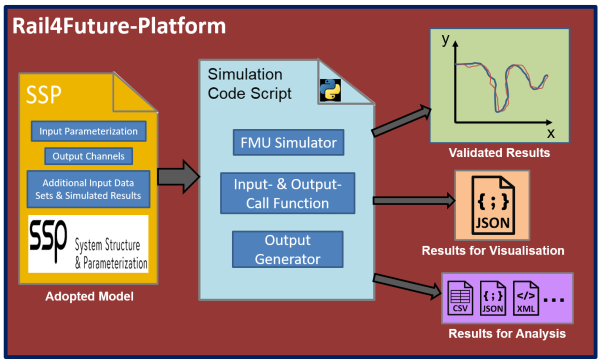


Figure 2. Model simulation approach.

After putting the SSP file into the platform, we provide a simulation code script, which can extract the FMU file from the SSP one, and all the data belonging to the FMU from the additional input data sets and simulated results with an input call function after executing it. After that, the code execution can put the FMU into simulation with an FMU simulator, then extract outputs from the simulation with an output call function, and finally generate all the outputs with an output generator in the platform. Surely, the functional components of the script should have the necessary software tool, packages and libraries, which can be provided from their official web pages or publicly available repositories freely as found out for this research work before. In the R4F Platform, the simulation code script can be executed by the Jenkins pipeline technology as we tested with the two previously mentioned use cases before, which is the key success of this work to integrate the models of different railway infrastructure subsystems into the platform.

The final step is to obtain the results of the entire simulation in order to analyse, validate and finally visualise them in the platform for the end user. First, the results of the SSP simulation are compared with the simulated results, generated from the simulation in the default software tool, in a plot (Validated Results). Second, the results to be visualized are converted into JavaScript Object Notation (JSON) format for visualization to the end user, as this open standard file format, with its human-readable key-value paired data structure, is well suited to the visualization part of the R4F platform. Finally, the results can be generated in different file formats (e.g. CSV, JSON, XML...), which the end user can analyze in different well-structured data types such as tables, arrays, lists, etc.

However, there are some challenges to be faced, namely dependencies and limitations, that we need to consider for the SSP simulation in the platform. For example, valid commercial license servers, Virtual Private Network (VPN) in some cases, OS-dependency of the FMU file and some open-source tools with necessary libraries and packages need to be connected and provided in the platform to make this kind of simulation successful. Therefore, it is essential have knowledge and experience of the model simulation process, data management, software installation, configuration and license management. Furthermore, the importance of dealing with IT-specific problems (e.g. system crashes, unsuccessful command executions) cannot be denied as we have realized specifically for this research.

4 Use Case Examples

In this section, first, we define the two use cases: 1) RLT calculation of a steel bridge and 2) MBS model of a railway vehicle. Second, we explain the conversion of their models into the FMU- and SSP-formats in detail. Then we demonstrate the simulation process of their SSP-standardized models in the R4F Platform. Lastly, we compare their SSP-simulation results and results provided by their default simulation tool with some plotted figures for validation purposes.

4.1 Use Case 1: Residual Life Time Calculation of a Steel Bridge

The use case demonstrates the life time and damage sum calculation of a steel bridge by using a Python simulation model with its input files containing demo data such as train data, influence lines, bridge positions and detail categories. For this work, the whole asset of the use case was provided by AIT - Austrian Institute of Technology GmbH. Validation of the structural life time with respect to fatigue failure is done for five different details of the designed bridge, visually shown by addressing their positions with points in Figure 3, which is provided by VRVis Zentrum für Virtual Reality und Visualisierung Forschungs-GmbH. Each of the detail positions is assigned a detail category representing the cyclic stress $\Delta\sigma_c$ in N/mm² corresponding to $2 * 10^6$ load cycles of the fatigue strength curve as given in EN1993-1-9. Detail categories and the X, Y and Z coordinates of the detail positions are given in Table 1.

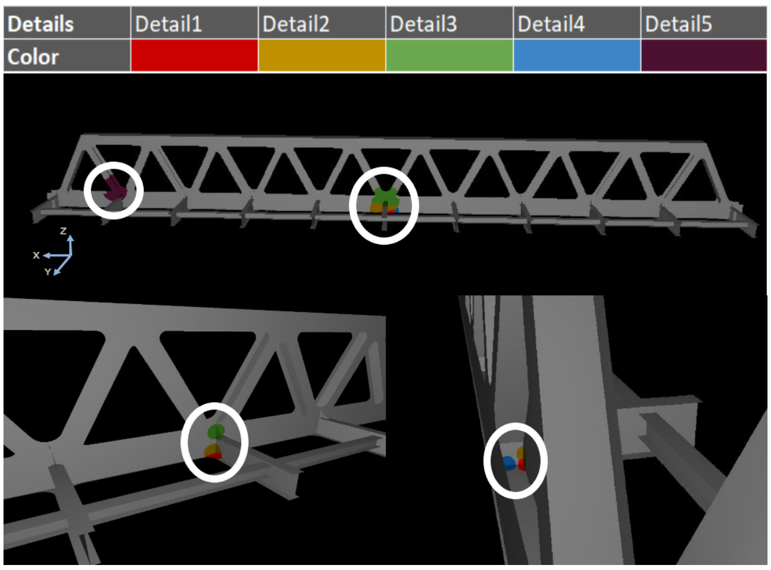
Table 1. Detail categories and positions of the steel bridge.

Detail	*Category [N/mm2]*	*X*	*Y*	*Z*
Detail1	80	0,057	0,306	-0,332
Detail2	80	0,009	0,265	-0,174
Detail3	36	0	0,32	0,56
Detail4	90	0	0	-0,332
Detail5	90	17,107	-0,088	0,866

In this section, first, we explain how the FMU and SSP formats are exported from the whole Python simulation model of the RLT calculation algorithm. Second, we demonstrate the entire simulation process of the use case using the model simulation approach mentioned before. Lastly, we compare the Python- and SSP-simulation results of the use case to each other by showing two diagrams for the result validation.

4.1.1 FMU Export

As the first step, we need to understand the Python simulation model consisting of different mathematical formulas with all its simulator, input and output components and bindings. After successfully testing the model simulation, we need to pack the model into the FMU format for co-simulation, which is possible by using the pythonfmu package (Hatledal, H. Zhang, and Collonval 2020). The next step is to prepare the model in a different Python code including all the formulas, input, output call functions and output generators in a pre-defined and imported pythonfmu class called FMI2Slave (`https://pypi.org/project/pythonfmu/`). Lastly, we execute a pythonfmu build command calling the prepared code to get the FMU format of the RLT calculation algorithm. In the FMU file, two different dll files in the bina-



Details	Detail1	Detail2	Detail3	Detail4	Detail5
Color	🟥	🟨	🟩	🟦	🟪

Figure 3. The steel bridge with highlighted detail points.

ries folder are found, which shows the OS-compatibility of the file in two different OSs (Linux & Windows).

4.1.2 SSP Export

After achieving the FMU file, we need to pack the FMU file into the SSP format to bring it into the R4F Platform as a complete model package with parameter description. For this purpose, we use the Model.CONNECT^{TM}, a commercial system integration tool provided by AVL List GmbH, because the tool can import FMU(s) for co-simulation and export SSP including the FMU(s). The SSP file also contains one SSD file describing and reserving all the registered parameters incl. inputs and outputs from the FMU-packaging process of the model.

4.1.3 Simulation Process

Figure 4 demonstrates the entire simulation process of the use case in a virtual machine, a prototype of the R4F Platform, due to the model simulation approach. First, we convert the prepared Python code with pythonfmu into the FMU format and lastly we achieve the SSP format of the model as mentioned in the previous subsection. From the SSP file, the FMU file is extracted and then simulated by running a simulation code script, which we wrote in the open-source Python programming language. Most importantly, the language supports the FMU-simulation for the use case by using the Python library called fmpy (`https://pypi.org/project/FMPy/`), which includes input and output call functions. In the use case of this work, the simulation code is actually the integrated version of the Python simulation of the prepared code included at the beginning. Moreover, we use a 2D-graphics package called matplotlib (Hunter 2007) in the simulation code to create static, interactive and publication quality plots. Additionally, we apply json and csv packages to the code to generate output in CSV and JSON formats after the simulation code script is executed by running the Jenkins pipeline. As outputs, the simulation gives out two plots from the matplotlib for result validation, CSV and JSON files for result analysis, and eventual visualization to the end user after reading all the provided CSV input files and Python-calculated life times and damage sums, which are the simulation results of the Python simulation model provided at the beginning.

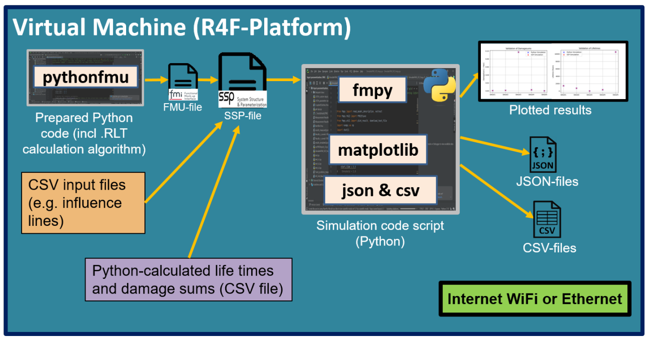

Virtual Machine (R4F-Platform)
pythonfmu
Prepared Python code (incl .RLT calculation algorithm)
FMU-file
SSP-file
fmpy
matplotlib
json & csv
Simulation code script (Python)
Plotted results
JSON-files
CSV-files
CSV input files (e.g. influence lines)
Python-calculated life times and damage sums (CSV file)
Internet WiFi or Ethernet

Figure 4. Simulation process of the RLT bridge.

4.1.4 Comparison Analysis for Result Validation

Figure 5 shows two plots, where the results of both Python- and SSP-simulations (life times and damage sums) are compared to each other due to the five defined detail categories.

As realized from the result consistency between the Python- and SSP-simulation in Figure 5, the SSP-simulation works properly and is suitable to be integrated into the R4F Platform.

4.2 Use Case 2: Multibody Simulation of a Railway Vehicle

The use case demonstrates the drive of a railway vehicle on a track. For this work, we use a generalized MBS model of the vehicle, which is parameterized, based on the Manchester Benchmark (Iwnicki 1999) and provided by Virtual Vehicle Research GmbH (ViF). We analyze the model in Simpack version 2022x, a commercial software tool from Dassault Systèmes, that provides a reliable way of understanding the dynamic behaviour of the railway vehicle due to the interaction between the vehicle and track. In the interaction, the irregularities of the track geometry, causing dynamics forces on the vehicle and track, are certainly to be taken into account in the MBS, because the track irregularities can lead to potential wheel damage and, in severe cases, derailments.

In this section, first, we explain the FMU- and SSP-export of the model due to the model standardization approach mentioned before with necessary methods, tools, inputs and outputs. After that, we describe the simulation process of the SSP-standardized model in a figure and finally compare its results with the Simpack simulation outputs resulted from the same input parameters.

4.2.1 FMU Export

After understanding and testing the whole simulation process of the MBS vehicle in Simpack, we need to define the necessary input parameters and output channels used for the FMU export. In the case of this work, first, we set up the track irregularities previously. Second, we define four different scenario parameters to realize a curved / straight vehicle drive with / without passengers for the input parameterization: 1) Track curve radius; 2) Track superelevation; 3) Additional mass (passenger+luggage); 4) Vehicle speed (constant). Besides, eight output channels of the vertical deflection of the primary spring (4 for each one of the two bogies) and one of track position are registered to the MBS model, because they are used for fatigue life calculation of the primary spring and therefore shown in a graph. Then we can directly obtain the FMU format of the model, including the Simpack solver, by using the Simpack tool. For this work, we export the MBS model into FMI 2.0 for Co-Simulation, which is the latest FMI version supported by the Simpack 2022x. Furthermore, we discovered that the FMU file is dependent on the OS, where the Simpack 2022x was installed and the

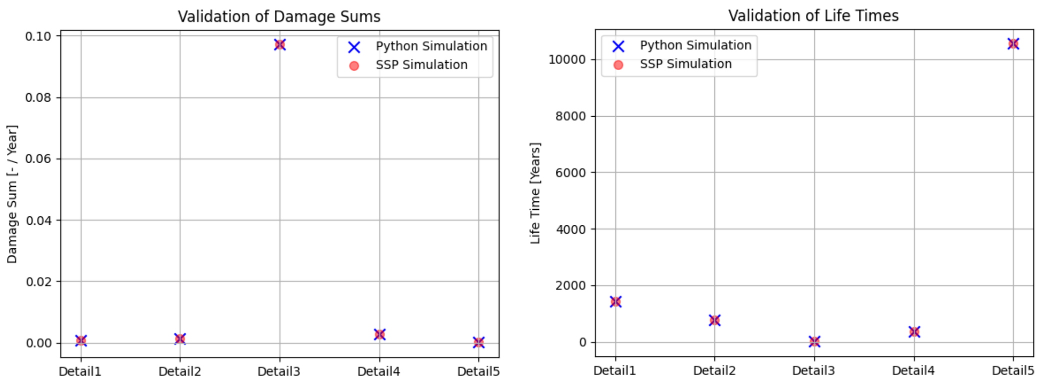


Figure 5. Simulation result comparison based on demo data of the RLT bridge.

file was exported from the tool before. It means that the file has only one dll file, working for the one OS, in the binaries folder. Therefore, the OS of the R4F Platform must definitively be taken into account to realize the simulation of the model in the platform.

4.2.2 SSP Export

After creating the FMU format of the MBS model, we need to convert it into the SSP format and then simulate it in the R4F Platform. For the task of this work, we use the same tool Model.CONNECT^{TM}.

4.2.3 Simulation Process

After achieving the SSP-standardized model, it is simulated by running the Jenkins pipeline in the platform in a proper way. Therefore, we need to build, design and optimize the simulation process of the model based on the proposed model simulation approach mentioned above. After that, we need to adapt the whole process to the platform.

In Figure 6, we describe the considered simulation process of the MBS model with useful software, interfaces, file formats, software packages, adapters, inputs and outputs in the virtual machine. First, the FMU file in the SSP is read by a simulation code script after the FMU- and SSP-export of the MBS model. We wrote the script in the open-source Python language as the one in the previously mentioned use case. We apply the fmpy package for the FMU-simulation of the use case in the script as well. In addition, we use a Python library called bs4 in the code script, because it helps to call inputs directly from the SSV of the SSP file, where the end user can see and configure the clearly shown input parameters before starting the model simulation. Besides, like in the previously mentioned use case, we propose to use the matplotlib, json and csv packages in the script as output generators, which enable to create JSON file(s) for visualization, CSV for analysis and 2D-plot(s) to validate the SSP simulation with the Simpack simulated data, extracted from the previous Simpack simulation of the MBS model, in CSV and/or TXT format. Additionally, we use the Command Line Interface (CLI) technology to allow the end user to enter their inputs directly into the CLI window for demonstration purposes as an alternative to the input parameterization with the SSV file mentioned before.

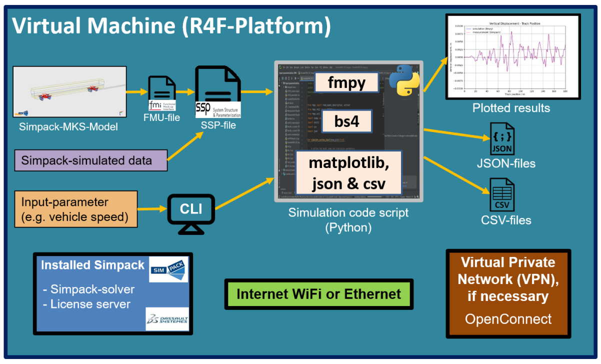


Figure 6. Simulation process of the MBS vehicle.

Furthermore, to overcome difficulties and limitations such as the connection to an available Simpack license server (either with VPN or without), OS-dependency of the FMU and internet availability in the simulation process, we installed and implemented the Simpack tool, a VPN service provider (e.g. OpenConnect tool) and an additional license server file, checked by the Simpack server to verify the licensing, in the platform.

4.2.4 Comparison Analysis for Result Validation

After running the simulation of the SSP-standardized model in the platform, we need to compare the Simpack- and SSP-simulation results to each other to validate the functionality of the SSP file in the proposed simulation process. In Figure 7, we show the vertical deflection of one primary spring, extracted from one of the previously mentioned output channels, due to the track position of the railway vehicle, which is loaded (passengers+luggage) and driven with a constant speed of 120 km/h on a tangent track without superelevation (straight drive) as an example.

As realized from the result consistency between the Simpack- and SSP-simulation in Figure 7, the SSP-simulation works properly and is suitable to be integrated into the R4F Platform.

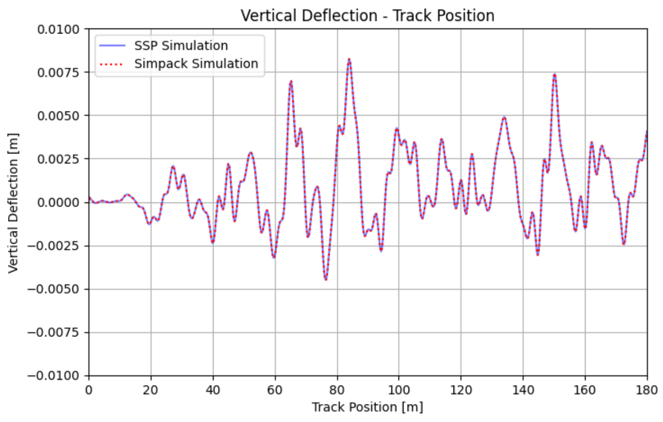


Figure 7. Simulation result comparison of the MBS vehicle.

5 Proposed Integration Process

Figure 8 shows the proposed integration process to be applied to the R4F Platform, which helps to manage and maintain the communication, interaction of the FMI- and SSP-standardized railway infrastructure models, provided from the Asset Provider, their final visual representation and enhanced user control for the end user properly. First, to continuously integrate the models in the platform, we propose to use the Jenkins pipeline technology in this research paper as mentioned before. Second, we will apply a version control system to the platform to manage the configuration, storage, exchange and archiving of necessary files and source codes related to the two use cases (SSP file, simulation code script, input & output files, pipeline code) in a software repository collaboratively. Besides, we discovered the Source Code Management (SCM) system in the Jenkins pipeline, which helps the pipeline to extract the updated sources from the repository. The reversed process is also possible by pushing the new output files back to the repository for result analysis as done by executing particular useful commands in the pipeline code before. Furthermore, we will use the pipeline auto-generation technology with a graph DBMS in the platform as mentioned previously (see (Reiterer, Schiffer, and Benedikt 2022) for example). In this research work, we created and configured a Jenkins pipeline, then described and modeled its workflow by developing a simple control script and a pipeline graph for testing purposes. Lastly, a visualization prototype is under development which demonstrates visualization and interaction techniques for the R4F Platform according to the use cases. It will allow users to set simulation input parameters and explore simulation results. This is done using a combination of linked views like 2D charts, 3D views, maps, and specially designed visualizations for exploring simulation results in a spatio-temporal context.

As we experienced in the model integration process of the two previously mentioned use cases before, there are some dependencies and limitations such as license server, VPN, internet and OS-dependency to overcome for the accomplishment of the integration process in the R4F Platform. For this research work, the Asset Provider exports the FMI format from various tools (e.g. pythonfmu, Simpack), SSP format directly from the Model.CONNECT^{TM} tool and essential Python packages from the Python tool other than providing the railway submodels to the platform. Besides, we surely need to do some internal configurations in the platform such as connecting the platform to a license server (e.g. Simpack license server), providing the OS, internet WiFi and/or Ethernet for wireless/wired network connection, 5G for mobile broadband network connection with relatively high internet speed and VPN service client for the license server connection in some cases.

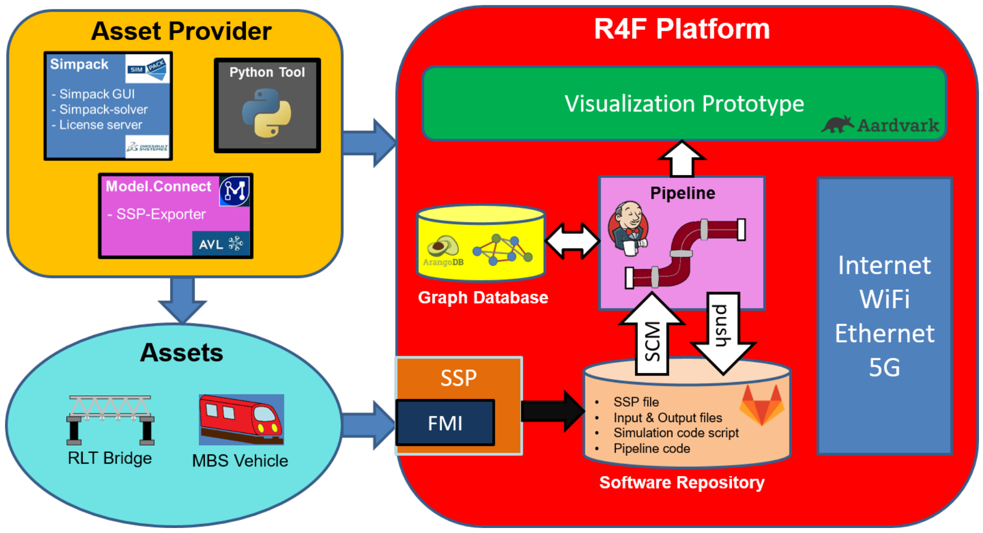


Figure 8. Proposed integration process.

6 Conclusion and Outlook

On the basis of the relatively precise result consistencies shown in the two use cases, the FMI- and SSP- standards are promising to be applied to different railway submodels for their integration into the platform. As experienced in this work, the FMI technology is a potentially useful interface standard supported by many different tools to adapt the models to the platform in terms of tool-independency, platform-compatibility and ease-of-use (portable as file). Moreover, the SSP-package of the models shows great benefits to enhance the IP protection, clear description, storage and exchange of different parameters belonging to the models. In addition, the SSP simulation of both use cases is carried out by executing the Jenkins pipeline interacted with a software repository belonging to a version control system, which also shows high potential in adaptiveness of the simulation to the R4F Platform.

Of course, there are more challenges to overcome to fully realize the model integration and interoperation in the platform. In the future, different use cases should interoperate with each other and be further described by using the FMI and SSP technologies. For example, a surrogate model (machine learning with the MBS model), proposed by (Zhou, Meierhofer, et al. 2023) to reduce the computational effort of the traditional MBS, is going to be applied to the platform by using both suggested interface standards. Additionally, an anti-slip control system is planned to be integrated into the platform as co-simulated with the MBS model to be able to co-accelerate and co-brake the railway vehicle. Finally, the proposed integration process is going to be further developed and optimized in a virtual machine by using containerization, container orchestration and deployment technologies, which assist to integrate and deploy the FMI-standardized models and their belonging data continuously, time- and energy-sustainably as a major benefit in future.

Acknowledgements

The authors would like to acknowledge the financial support of the COMET project Rail4Future (882504) within the COMET Competence Centers for Excellent Technologies from the Austrian Federal Ministry for Climate Action (BMK), the Austrian Federal Ministry for Digital and Economic Affairs (BMDW), the Vienna Business Agency and the Styrian Business Promotion Agency (SFG). The Austrian Research Promotion Agency (FFG) has been authorised for the COMET program management. We further thank for the support from ÖBB-Infrastructure AG and Siemens Mobility Austria GmbH.

References

Ahmadi, Miad et al. (2021). “Adapting digital twin technology in electric railway power systems”. In: *2021 12th Power Electronics, Drive Systems, and Technologies Conference (PEDSTC)*. IEEE, pp. 1–6.

Balakirsky, Stephen et al. (2010). *Simulation, Modeling, and Programming for Autonomous Robots*. Springer.

Bastian, Jens et al. (2011). “Master for co-simulation using FMI”. In: *Proceedings of the 8th International Modelica Conference; March 20th-22nd; Technical Univeristy; Dresden; Germany*. 63. Linköping University Electronic Press, pp. 115–120.

Bertsch, Christian, Elmar Ahle, and Ulrich Schulmeister (2014). “The Functional Mockup Interface-seen from an industrial perspective”. In: *Proceedings of the 10 th International Modelica Conference; March 10-12; 2014; Lund; Sweden*. 096. Linköping University Electronic Press, pp. 27–33.

Blockwitz, Torsten et al. (2012). “Functional mockup interface 2.0: The standard for tool independent exchange of simulation models”. In: *Proceedings*.

Golightly, David et al. (2022). “A feasibility assessment of multi-modelling approaches for rail decarbonisation systems simulation”. In: *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit* 236.6, pp. 715–732.

Hällqvist, Robert et al. (2021). “Engineering domain interoperability using the system structure and parameterization (ssp) standard”. In: *Modelica Conferences*, pp. 37–48.

Hamarat, Mehmet, Mayorkinos Papaelias, and Sakdirat Kaewunruen (2022). “Fatigue damage assessment of complex railway turnout crossings via Peridynamics-based digital twin”. In: *Scientific reports* 12.1, p. 14377.

Hartmann, Bjørn (2020). “Digital Twin Monitoring of Offshore Knuckle Boom Crane”. MA thesis. NTNU.

Hatledal, Lars Ivar, Houxiang Zhang, and Frederic Collonval (2020). “Enabling Python Driven Co-Simulation Models With PythonFMU.” In: *ECMS*, pp. 235–239.

Hunter, J. D. (2007). “Matplotlib: A 2D graphics environment”. In: *Computing in Science & Engineering* 9.3, pp. 90–95. DOI: 10.1109/MCSE.2007.55.

Iwnicki, SD (1999). *The Manchester Benchmarks for Rail Vehicle Simulation. Department of Mechanical Engineering, Manchester Metropolitan University, England, Supplement to Vehicle System Dynamics, Vol. 31, ISSN 0042–3114*.

Kaewunruen, Sakdirat, Mohannad AbdelHadi, et al. (2022). “Digital Twins for Managing Railway Bridge Maintenance, Resilience, and Climate Change Adaptation”. In: *Sensors* 23.1, p. 252.

Kaewunruen, Sakdirat and Qiang Lian (2019). “Digital twin aided sustainability-based lifecycle management for railway turnout systems”. In: *Journal of Cleaner Production* 228, pp. 1537–1551.

Kaewunruen, Sakdirat and Ningfang Xu (2018). “Digital twin for sustainability evaluation of railway station buildings”. In: *Frontiers in Built Environment* 4, p. 77.

Mysari, Sriniketan and Vaibhav Bejgam (2020). “Continuous Integration and Continuous Deployment Pipeline Automation Using Jenkins Ansible”. In: *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*. IEEE, pp. 1–4.

Ochel, Lennart et al. (2019). “Omsimulator–integrated fmi and tlm-based co-simulation with composite model editing and ssp”. In: *Proceedings of the 13th International Modelica Conference, Regensburg, Germany, March 4–6, 2019*. 157. Linköping University Electronic Press.

Pieper, Tobias and Roman Obermaisser (2018). “Distributed co-simulation for software-in-the-loop testing of networked railway systems”. In: *2018 7th Mediterranean conference on embedded computing (MECO)*. IEEE, pp. 1–5.

Pierre R. Mai (2018-10-11). *MAP “System Structure and Parameterization” – Current Status and Plans*. Modelica Association. URL: https://ssp-standard.org/publications/2018_American_Modelica_Conference/2018_Usermeeting_SSP-StatusandPlans.pdf (visited on 2023-05-12).

Reiterer, Stefan H, Clemens Schiffer, and Martin Benedikt (2022). “A Graph-Based Metadata Model for DevOps in Simulation-Driven Development and Generation of DCP Configurations”. In: *Electronics* 11.20, p. 3325.

Thule, Casper et al. (2018). “Towards the verification of hybrid co-simulation algorithms”. In: *Software Technologies: Applications and Foundations: STAF 2018 Collocated Workshops, Toulouse, France, June 25-29, 2018, Revised Selected Papers*. Springer, pp. 5–20.

Zhang, Shiyao et al. (2021). “A digital-twin-assisted fault diagnosis of railway point machine”. In: *2021 IEEE 1st International Conference on Digital Twins and Parallel Intelligence (DTPI)*. IEEE, pp. 430–433.

Zhou, Shiyang, Stefan Dumss, et al. (2022). “A Conceptual Model-based Digital Twin Platform for Holistic Large-scale Railway Infrastructure Systems”. In: *Procedia CIRP* 109, pp. 362–367.

Zhou, Shiyang, Alexander Meierhofer, et al. (2023). “A Machine-Learning-based Surrogate Modeling Methodology for Submodel Integration in the Holistic Railway Digital Twin Platform”. In: *Procedia CIRP* 119, pp. 345–350.