Pipeline-based Automated Integration and Delivery Testing of Simulation Assets with FMI/SSP in a Railway Digital Twin

Ozan Kugu[1] Shiyang Zhou[1] Stefan H. Reiterer[2] Mario Schwaiger[2] 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]Virtual Vehicle Research GmbH, 8010 Graz, Austria, stefan.reiterer@v2c2.at

Abstract

Railway infrastructure systems have recently been enhanced through the use of the digital twin (DT) concept, enabling visualization and control in a virtual environment while effectively mitigating life cycle costs. This work provides insights into the development and operations (DevOps) of a railway DT platform and highlights the automation and management of asset integration and processing based on the FMI and SSP interface standards through the use of the Continuous Integration / Continuous Delivery pipeline technology. This offers long-term durability, pausability, remote triggering, open-source and workflow design capabilities, and connectivity to other tools such as version control systems and code analysis tools. In this research paper, we present an anti-slip co-simulation model of a railway vehicle as a use case example to demonstrate the pipeline-oriented automation and management in combination with a version control system and code analysis tool within the platform.

Keywords: CI/CD Pipeline, DevOps, FMI, SSP, Automation and Management, Asset Integration and Processing

1 Introduction

Railways play a crucial role in modern public and freight transportation due to their cost-effectiveness, energy-efficiency and eco-friendliness. To reduce life-cycle costs, conserve energy, and streamline maintenance and monitoring for railway operators and infrastructure managers, railway infrastructure systems are being brought into the virtual world through the use of the DT concept. As an example, (Zhou et al. 2022) conceptualized a DT platform called Rail for Future (R4F), where digital assets (models and data) from different railway subsystems, including vehicles, tracks, turnouts, bridges and tunnels, can be integrated and interoperated with each other. This enables end-users to control and visually analyze the railway system.

There are challenges and limitations to overcome in railway digitalization. For instance, raw simulation assets of the subsystems cannot be run and easily managed in the DT platform due to their software tool dependence, operating system (OS) incompatibility, relatively complex model and data structure. The use of interface standards, offered by Modelica Association, shows great potential for dealing with the adaptation of the simulation assets to the platform. Some of these standards are Functional Mock-up Interface, which provides an interface between dynamic simulation models and software as a ZIP-formatted container (simulation unit called Functional Mock-up Unit (FMU)), is open-source and supported by more than 200 tools (*Functional Mock-up Interface (FMI) Standard* 2024), and System Structure & Parameterization, which is used to containerize complex simulation systems containing one or more FMUs and ideal for co-simulation use cases (*System Structure and Parameterization (SSP) Standard* 2024). This adaptation process allows the assets to be seamlessly integrated and processed in the platform. (Kugu et al. 2023) successfully demonstrated this in their work by using the FMI and SSP standards. In order to ease the asset integration and processing task, we prefer to automate it and improve its management. This is a challenging mission, that requires comprehensive knowledge and experience about the assets, the task and automation techniques to apply it to the platform. This paper demonstrates the use of the Continuous Integration / Continuous Delivery and/or Continuous Deployment (CI/CD) pipeline technology for automated integration, processing and management of the assets with the FMI and SSP standards in the R4F Platform. The technology is foreseen as an effective method for the automation, because it provides workflow design, open-source, delivery and/or deployment capabilities, remote-triggering functions, pausability (ability to stop and wait for human response), long-term durability and platform compatibility. Besides, the pipeline software tool can interoperate and communicate with other software tools such as version control systems and code analysis tools, which help track and store the version of the assets and detect potential errors, vulnerabilities and redundancies in the codes belonging to these assets. Thus, version control systems and code analysis tools are applied to the pipeline in this work, which effectively boosts the automation and management of the asset integration and processing task as a part of the DevOps practice in the platform. Moreover, there are open-source tools available for these two tech-

nologies. Considering all the features and open-source materials, mentioned above, the CI/CD pipeline technology is highly preferred for this research work.

Another issue to address concerns license management, which is necessary to obtain permissions for running simulations of the assets in the pipeline of the platform, since these assets were previously designed and configured with an appropriate solver in commercial software tools. Additionally, a thorough understanding of pipeline installation, configuration, and design is needed, which also depends on the physical domain according to (Zampetti et al. 2023)'s comprehensive study on CI/CD pipeline implementation in a DT for Cyber Physical Systems (CPS) incl. railways. Moreover, proficiency in installing, configuring version control systems and code analysis tools, and effectively interoperating the pipeline with them in the platform is essential.

In this research paper, first, in Section 2, we give insights about the benefits and limitations of railway DTs, CI/CD pipeline technology, FMI and SSP standards through different research examples. Second, in Section 3, we define the FMI- and SSP-based asset integration and processing task and its underlying goals. Then, in Section 4, we present what kind of advantages the automation and management of the task bring to stakeholders, how we apply it to the R4F Platform, how we keep the stakeholders updated through the use of the pipeline, version control system & code analysis tool methodologies and which processes we defined and directly applied to the pipeline in the platform. After that, in Section 5, we show an interesting demonstration of how we automate and manage the integration and processing of a co-simulation model consisting of a multibody simulation (MBS) model of a railway vehicle, and a Proportional-Integral-Derivative (PID) controller model for anti-slip traction and vehicle speed control of the vehicle. In Section 6, we discuss the simulation comparison results of the use case, and point out limitations and challenges we encountered while integrating, processing the assets, then automating and managing them in the pipeline. Finally, in Section 7, we briefly mention the conclusion of this work and outline future work.

2 Related Work

2.1 Railway Digital Twin

In recent years, railway DTs have increasingly been designed and developed by numerous researchers and engineers to enhance operational, monitoring and maintenance tasks within the virtual railway environment. For instance, (Zhang et al. 2021) presented a DT-assisted approach for fault diagnosis of railway point machines used to operate turnouts. They emphasized the significance of DTs in enhancing fault diagnosis processes, thereby improving the reliability and efficiency of railway operations. In 2022, (Hamarat, Papaelias, and Kaewunruen 2022) introduced a Peridynamics-based DT approach, which could predict potential fatigue damage in railway turnout crossings by integrating real-time data and simulations, facilitating proactive maintenance and improved safety. This enhancement contributed to assessment and management of railway infrastructure. Additionally, (Kaewunruen et al. 2023) proposed employing DTs for managing railway bridge maintenance, where they monitor and analyze the structural integrity of railway bridges in real time. Their study highlighted the role of DTs in improving decision-making processes related to the maintenance, thereby enhancing the overall safety and reliability of railway infrastructure systems. As mentioned in Section 1, (Zhou et al. 2022) conceptualized a model-based DT platform called R4F, capable of simulation, visualization, and predictive analytics, enabling stakeholders to optimize operations, maintenance, and resource allocation for comprehensive management of large-scale railway infrastructure systems. This advancement improved the efficiency and reliability of the system. This paper aims to ease stakeholders' aforementioned tasks by demonstratively automating and managing the integration and processing of various railway simulation assets within the platform.

2.2 CI/CD Pipeline for Digital Twins

Many researchers and software engineers prefer CI/CD pipelines to automate and manage integration, simulation, validation, delivery and deployment processes in a DT of a physical system as DevOps practices. This approach facilitates easier analysis, monitoring, and evaluation of the system. For example, (Hugues et al. 2020) proposed the TwinOps process, which is a combination of DevOps, DTs and model-based engineering, and used it for automated code generation, condition monitoring and data analysis of CPSs. (Villa et al. 2024) used the CI/CD pipeline technology to reproduce protocol stacks (e.g. cellular, WiFi) in both physical and digital environments in real time, which helps researchers to efficiently and automatically test the protocols in a conceptual DT for large-scale wireless networking. (Barbie, Hasselbring, and Hansen 2023) enhanced the automated testing of their DT prototype for smart farming applications through the use of the pipeline technology. They noted relatively high cost and time consumption of the hardware used for the applications, making simulations a preferable choice over the hardware for gaining virtual insights into the physical system via the CI/CD pipeline. Consequently, we aimed to work with multiple railway simulation assets, intending to automatically integrate and process them within our CI/CD pipeline for this work.

Another crucial aspect to consider is the intercommunication of the pipeline with other tools, which helped us to further improve the management of our automated asset integration and processing task occurring in the pipeline. For instance, (Kiran et al. 2021) suggested to work with code analysis tools in their CI/CD pipeline for more reliable and secure software development life cycle and DevOps. Similarly, (Zampetti et al. 2017) did a relatively extensive comparative study, where they investigated the us-

age of several static code analysis tools in a CI pipeline for static analysis of many different open-source software applications. They also gave many insights about how effectively the static code analysis tools should be used with the CI pipeline to detect bugs, errors and warnings in the application examples, which is important for us to be aware of possible failures, redundancies and vulnerabilities in our asset applications. Besides, (Sethi 2020) proposed to apply version control system to their CI/CD pipeline for business intelligence solutions in order developers to keep tracking and saving source code changes collaboratively. Based on this, we found the version control system very promising for tracking and storing our asset applications with a version control system tool directly connected to our CI/CD pipeline for this work.

2.3 FMI and SSP Standards for Railways

The FMI and SSP standards are preferred to be used in many sectors including the railway sector, because these strongly assist researchers to integrate and manage different railway simulation models in a virtual environment by providing more tool-independence, file-portability, co-simulation capability and less data complexity in spite of particular limitations noted by railway experts.

(Pieper and Obermaisser 2018) introduced a distributed co-simulation approach for conducting software-in-the-loop tests of networked railway systems. In this approach, various subsystems of the railway network could be simulated independently and in parallel by leveraging FMI. They also highlighted the potential of FMI in enabling collaborative and scalable simulation of the systems, which is crucial for ensuring their reliability and safety in operations. (Hotzel Escardo et al. 2021) designed and developed a train driver behaviour model for railway co-simulations, demonstrating how train driver behavior models can be seamlessly integrated with other simulation components, such as infrastructure and rolling stock models by utilizing FMI. This approach allows for comprehensive simulations that capture the interactions between different elements of railway systems, ultimately enhancing the understanding of system dynamics, supporting decision-making processes in railway operations, and planning. Besides, (Zhou et al. 2023) proposed to use FMI to seamlessly integrate their machine learning based surrogate model in the R4F Platform. This enables tool-independency and interoperability with other future models. (Golightly et al. 2022) noticed several FMI-compliant simulation tools (e.g. MATLAB/Simulink), used for rail applications and thus showing great potential of the FMI standard for the rail sector, while they studied the practicability of the multi-modelling approach for rail decarbonisation systems. On the other hand, they listed a couple of limitations of the FMI for railways such as lack of clear presentation, data incompatibilities and intellectual property (IP) issues related to railway simulation models, which are surely to be considered while using FMI for railway applications as well. (Hällqvist et al. 2021) discussed the utilization of the SSP standard to achieve engineering domain interoperability, particularly focusing on its application within railway systems. The study highlights how SSP facilitates interoperability between various engineering domains, including railway systems, by providing a common standard for describing system components and their interactions. By adopting SSP, engineers can enhance collaboration, optimize model integration processes, and increase the efficiency of complex railway system development and analysis. Finally, (Kugu et al. 2023) proposed to use both FMI and SSP to integrate and simulate different railway simulation models such as an MBS model of a railway vehicle, and residual life time calculation model of a railway steel bridge in the R4F Platform. They also pointed out the high potential of these standards for the railway sector, which gave us enormous inspiration to use the FMI and SSP technologies for our asset integration and processing task in the CI/CD pipeline on the platform in this work.

3 Asset Integration and Processing

The asset integration and processing task plays a significant role in enabling various railway use cases to operate within the R4F Platform. This task comprises two primary components: Asset Integration and Asset Processing.

The Asset Integration in this work means direct adaptation of the simulation assets consisting of model and data to the platform through the use of interface technologies such as FMI and SSP. Before the Asset Integration part, these assets were manually designed and processed in a Graphical User Interface (GUI) - supported software tool (e.g. MATLAB, Simpack,...), which is very handy to create, configure, optimize and simulate a physical system. The system can be a railway vehicle as an example from the railway sector. Of course, these assets need to be further prepared to properly use it with the interfaces in the platform. In this preparation, input parameters, input and output channels are defined and assigned to the simulation assets. This is necessary to address the right inputs and outputs for the asset simulation, so that the input parameterization and output generation work in a right manner during the asset simulation executed by the pipeline in the platform. After that, these assets are packed into FMU and lastly into SSP, where the FMUs are connected to each other resp. co-simulated. This helps to achieve the assets as a container in ZIP format including the entire asset description (model metadata, parameters, connections, connectors). This container can technically be analyzed, simulated in the platform, and therefore makes the assets independent from their GUI-based default tool as found out in this work.

Asset Processing occurs subsequent to Asset Integration. In this case, the assets, integrated with FMI and SSP, are tested, optimized and then released to be completely sure that these function in a right manner. For this, necessary software tools, libraries, packages and licenses are

pre-installed on a computer as first step. After that, codes of the assets are manually analyzed to find out errors and then fix bugs. After the code analysis, these assets are simulated and finally their outputs are generated by executing a simulation code script on the computer (see the model simulation approach of (Kugu et al. 2023)). Prior to delivery, all aspects related to the simulation test undergo peer review by the Asset Integrator to evaluate the test and determine whether to release the assets.

4 Automation & Management

4.1 Benefits

The asset integration and processing are automated and managed by the Asset Integrator through the use of the pipeline, version control system and code analysis tool in the R4F Platform as previously mentioned in Section 1. The reason for this is that it brings significant advantages as follows:

- **Time-efficiency** in the asset integration and processing through the automation,
- **Money-saving**, because open-source tools are implemented for the automation and management work,
- Stakeholders need **less prior knowledge** about the asset integration and processing through the Asset Integrator's great contribution to the automation and management,
- **Long-term durable, pausable and workflow-based** asset integration and processing through the pipeline technology,
- **Better quality control** through analysis and validation processes,
- **Better control of the tracking and storage** of the simulation assets between different stakeholders by using the version control system, therefore **easier collaborative work** between them.

4.2 Environment Overview

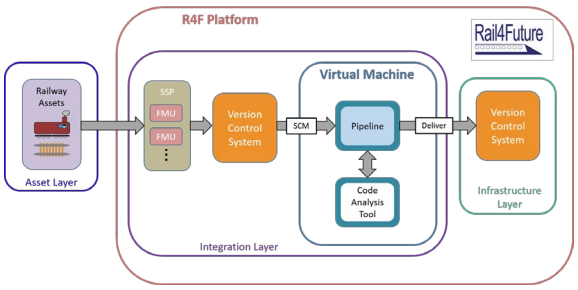


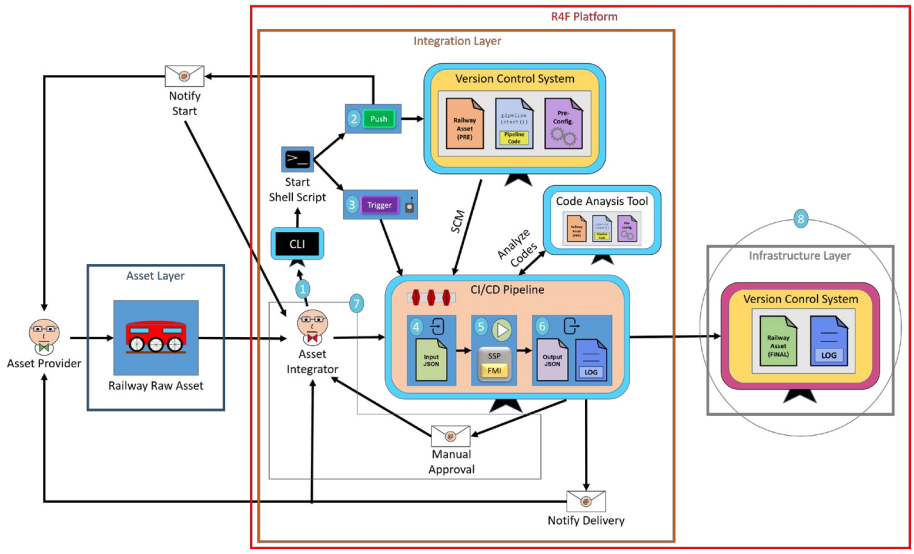
Figure 1. Simplistic Landscape of the CI/CD Pipeline-based Automation and Management of the Asset Integration and Processing in the R4F Platform.

Figure 1 provides an overview about how, in what kind of environment and among which layers of the R4F Platform landscape (as outlined in (Zhou et al. 2022)) the asset integration and processing task is automated and managed by using the CI/CD pipeline technology. As first step, the railway raw assets, which are provided from the asset layer into the integration layer of the platform, are adapted to the platform through the use of the FMI and SSP within the scope of the Asset Integration part. Then, as first of the Asset Processing part, these integrated assets are stored and tracked in the Asset Integrator's version control system, which is directly connected to a pipeline by using a Source Code Management (SCM) plugin in order the pipeline to keep and track these assets in its own server as well. Of course, the pipeline should be comparable with the ones in the R4F Platform function layer, where pipelines for particular functions (e.g., predictive maintenance) are used, because all the assets must work in the function layer's pipelines to be able to visualize and control these assets in the visualization layer of the platform. Thus, the Asset Integrator initiates a virtual machine (VM) with Linux OS and then installs a software tool where they created the pipeline. Besides, the pipeline is directly connected to a code analysis tool installed in the same VM, so that the tool extracts the codes of the assets from the pipeline, and then publishes the code analysis results to the Asset Integrator. In these results, they can detect potential errors, redundancies, bugs and vulnerabilities, which helps them to further improve the code quality of the assets in advance. Finally, the refined assets are delivered to the version control system of the R4F Platform infrastructure layer, where the Asset Integrator conducts simulations and oversees function layer pipelines through pipeline execution, subject to manual approval.

4.3 Followed Approach

Figure 2 reveals the exact methodology to realize the automation and management of the asset integration and processing task, where the Asset Integrator plays their main part, in the pipeline. First, they get the raw prototypical simulation assets from the Asset Provider as usual. After manually preparing these assets as mentioned in Section 3, the Asset Integrator uses their Command Line Interface (CLI) tool, which is very handy to execute commands without any GUI in background. In the CLI, they start a shell script, which is a text file and contains a sequence of commands to be executed for process automation in the CLI. As first step, this script pushes all the assets (pre-asset), configuration files (pre-configs), including necessary software libraries and packages for the asset simulation, and a pipeline code, defining the workflow of the pipeline, to the integration layer's version control system software repository by using the Git command (see *Git Documentation* (2024)). Besides, the shell script pushes notification to all the stakeholders via email by using an open-source mail transfer agent software to inform them about the pipeline execution start. One notification email is sent to the Asset Integrator and includes four links with corresponding port numbers belonging to the pipeline, code analysis tool, version control system and pipeline console output, helping them to directly monitor and handle the whole progress of the asset integration

10.3384/ECP207189 DOI



Main Procedure Followed	
1	Start the Shell Script
2	Push Pre-Assets and Notification
3	Remotely Trigger Pipeline
4	Provide Input for Simulation
5	Start the FMI-based SSP Simulation
6	Generate Results
7	Manually Approve the Delivery
8	Deliver Asset

Figure 2. Overview of the Automation and Management Environment.

and processing. Another email is redirected to the Asset Provider, and includes only the pipeline start message, not the four links due to the IP protection of the pipeline automation methodology. After the pushing step, the shell script triggers the pipeline remotely through the use of the pipeline server's token so that the pipeline starts automatically and extracts all the pre-asset, pre-configs and pipeline code through the SCM from the Asset Integrator's version control system. After that, the principal input parameterization, asset simulation and its output generation occur one after the other in the pipeline. In the meantime, all the codes related to the pre-asset are automatically reviewed by the Asset Integrator's code analysis tool. Then, the Asset Integrator checks the code analysis results, pipeline console outputs and simulation results, generated as curves and output files for data analysis and validation purposes, after getting the Manual Approval message with the pipeline link as an email automatically sent from the pipeline. It should be noted that the manual checking process can be fully automated in future as (Reiterer, Schiffer, and Schwaiger 2023) did Key Performance Indicator evaluation and quality check by using post-processing tools in their work. After they finish to adapt the simulation of the FMI- and SSP-standardized asset to the pipeline, they decide to deliver the asset as final asset, and its belonging log file (pipeline console output) directly to the infrastructure layer's version control system through another push command execution by the pipeline like the previous one in the shell script. If the pushing succeeds, a Delivery message is automatically sent to all the stakeholders via email in order to inform them about the asset delivery.

4.4 Designed Workflow of the Pipeline

In this subsection, the workflow, designed and applied to the pipeline through the use of the pipeline code, is further concretized and described by defining main processes and their sub-processes. First, we automatically integrate our asset application into the R4F Platform through the use of the version control system, FMI and a couple of Linux commands in the pipeline within the Build process in order to be able to test the application there. After the Build, in the Test process, we do semi-automated testing and validation of the whole application by using the FMI, SSP, code analysis tool and Linux commands in the pipeline for quality assurance of the asset simulation in the platform. If the simulation shows relatively high resilience, code quality and result consistency, the Asset Integrator decides to deliver the final asset to the infrastructure layer's version control system software repository, which happens in the Deliver process. Otherwise, the asset application requires further improvement and development, leading to the entire automated asset integration and processing starting again from the Build process.

4.4.1 Build

1) Checkout SCM: Connection of the Asset Integrator's version control system to the CI/CD pipeline through SCM.
2) Update Software: Update necessary software packages, libraries and tools for the asset simulation in the pipeline.
3) Build FMU: Automated packaging railway simulation assets into the FMUs.

4.4.2 Test

4) Analyze Codes: Scan source codes related to the assets with the code analysis tool.
5) Validate FMU: Check, if the FMUs work properly.
6) Gather Info FMU: Extract all metadata from the FMUs to display these data in the pipeline console output.
7) Update SSP: Load the FMUs into an example SSP file.
8) Simulate SSP: Test the FMI-based SSP co-simulation of the assets.

9) Generate Results: Publish results in different formats for data analysis and result validation.
10) Manual Approval: Notify the Asset Integrator about the completed asset simulation in order them to approve the asset delivery.

4.4.3 Deliver

11) Deliver Asset: Release the whole asset application to the infrastructure layer's version control system and then inform all the stakeholders about it as confirmation.

5 Use Case: Anti-Slip Traction & Vehicle Speed Control System

In the following use case, we established a traction control alongside a vehicle speed control for an existing MBS train model, having two bogies with four wheel sets (two wheel sets per bogie), based on the Manchester Benchmark (Iwnicki 1998), designed in the commercial software tool Simpack from Dassault Systèmes, and provided by Virtual Vehicle Research GmbH. This control is able to keep the longitudinal slip of each train wheelset on a constant user input value, while simultaneously controlling the vehicle speed based on a linear function with user input variables. The purpose of this use case is to enhance the extent of usage of the already existing train model by providing the option of simulating scenarios that are closer related to actual situations, such as accelerating and braking the vehicle for arrival and departure at different train stations.

For the anti-slip traction control, the MBS model was co-simulated with a control model based on a PID controller and designed in MATLAB/Simulink. It was established through the use of the SIMAT method, which serves as a co-simulation interface between the Simpack server and MATLAB client. This interface was implemented as a SIMAT block, representing the MBS model, and directly connected its input and output channels to the input and output ports of the PID controller model in Simulink. The controller model uses the current vehicle speed and four wheel speeds of the four wheel sets in the two respective bogies as inputs. By calculating the current slip and the deviation to the desired slip, which is processed by the PID controller, the controller model generates an output signal. This output signal then results in an additional torque on the axis of the respective wheel sets, thus braking or accelerating the wheels. The vehicle speed control was implemented as a polynomial 1st order MATLAB function, with acceleration input variable as the slope parameter and the initial vehicle speed input variable as the offset parameter. The eventual outputs of this control were the vehicle position, speed and acceleration, which were handed over to the MBS vehicle model as constraints for its speed control. (for more information about the physical structure and implementation of the co-simulation model see (Zhou et al. 2024))

To integrate this model into the R4F Platform we used an open-source library for Simulink called FMI Kit for Simulink (see *GitHub - FMI Kit for Simulink* (2024)). This facilitated the packaging of the PID-based Simulink controller model into an FMU file, which is directly executable in the platform. After defining the input and

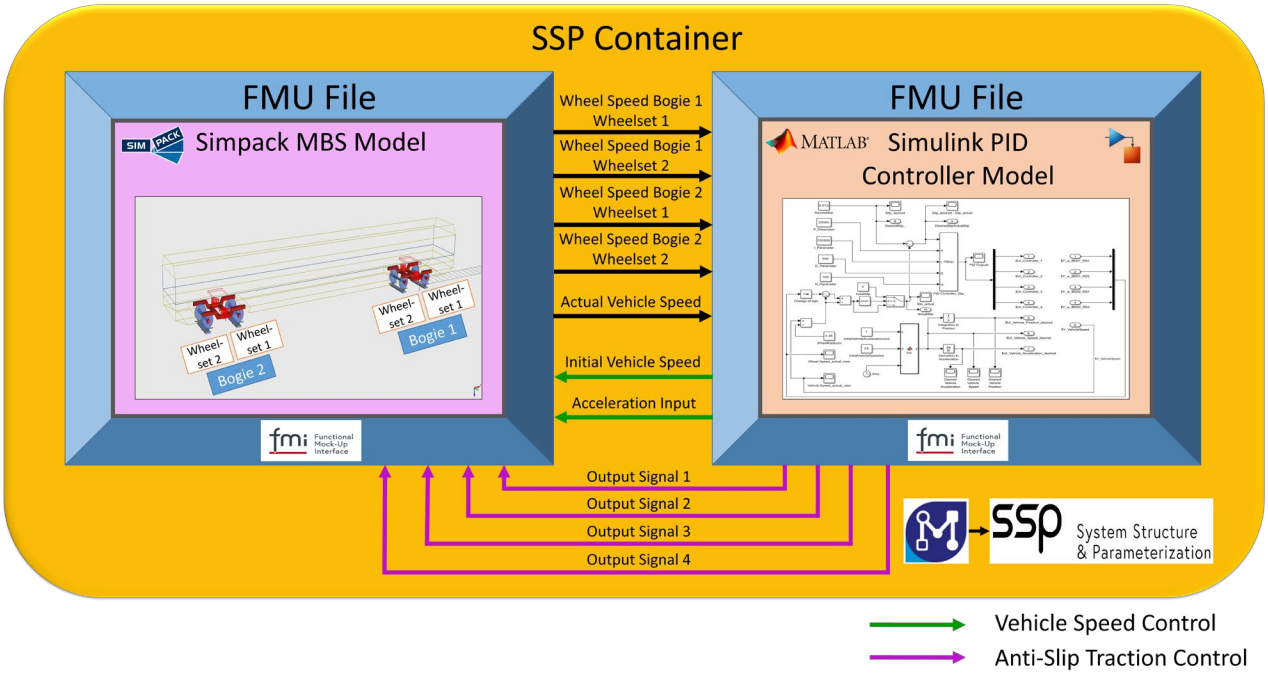


Figure 3. An Overview of the Physical Structure of the Anti-Slip Co-Simulation Model Adapted to the R4F Platform with FMI/SSP. (see (Zhou et al. 2024))

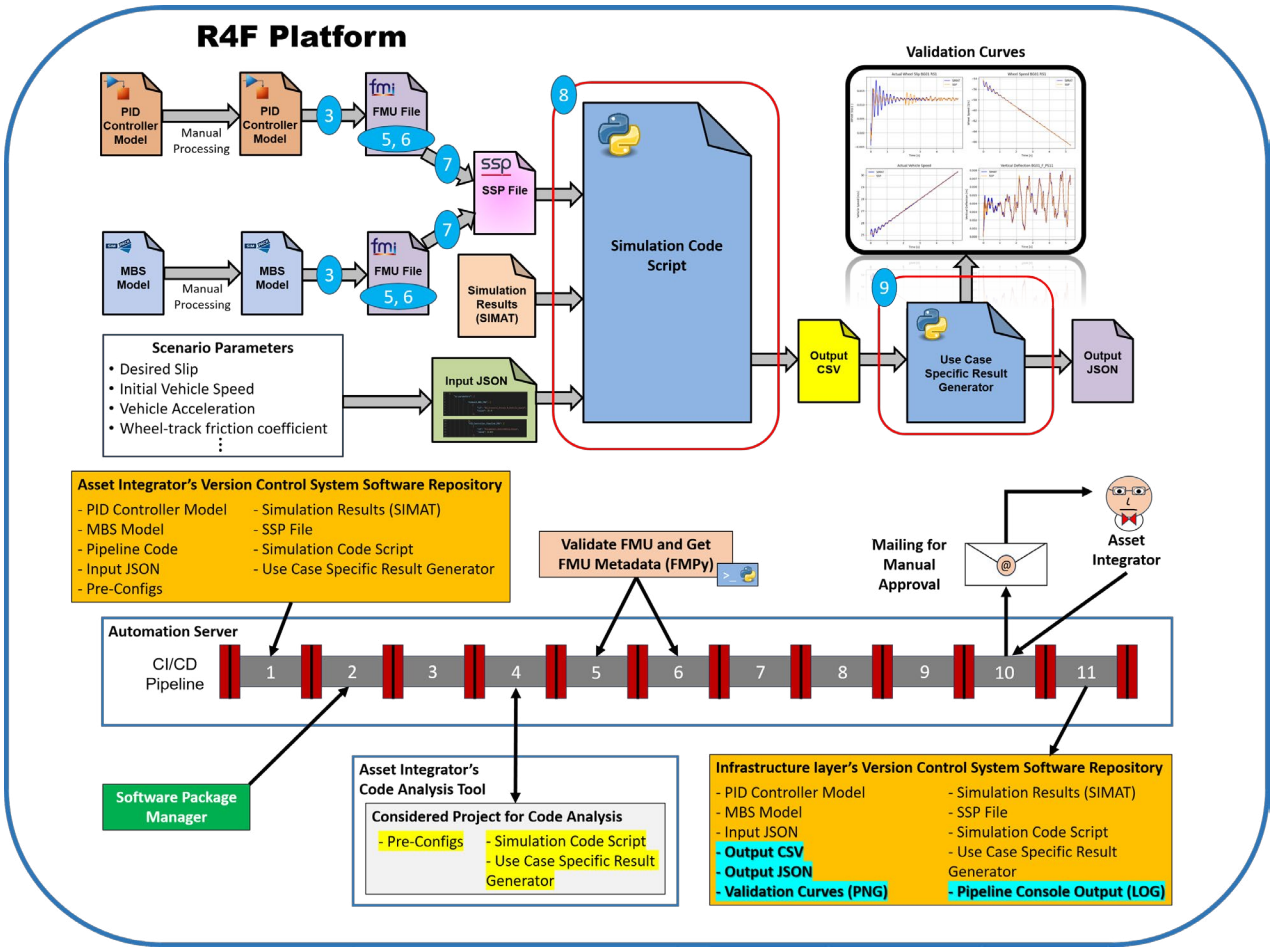


Figure 4. Automated Integration and Processing of the Anti-Slip Co-Simulation Model.

output channels of the MBS model in Simpack, we easily exported the model to the FMU by using Simpack itself. Then, we connected these two FMU files in Model.CONNECT, which is a co-simulation tool and provided from AVL List GmbH, to each other. Within the tool, we conducted tests to verify the co-simulation of these two models. After that, we used Model.CONNECT to pack the entire co-simulation model into an SSP example file, which contains the whole system structure and parameter description files with system metadata, connectors (input and output channels), connections and two FMU components belonging to these two models. Figure 3 shows an overview of the physical structure, input and output flow, described in the previous paragraph and occurring between the two FMU components in the SSP, of the entire co-simulation model.

Figure 4 illustrates the complete implementation of the use case in our CI/CD pipeline with the FMI, SSP standards, code analysis tool and version control system software repositories. Before the pipeline execution, first, the Simpack MBS model and Simulink PID controller model are manually processed by the Asset Integrator, by which scenario parameters (e.g., initial vehicle speed, desired slip, vehicle acceleration, etc.) given by the user are defined and configured. Besides, necessary input and output channels are created for their further interconnection and output generation. Following the manual processing, the pipeline extracts all asset files, including pre-configs, the PID controller model, MBS model, pipeline code, input JSON file, simulation results from the SIMAT, SSP example file, simulation code script, and use case specific result generator, from the Asset Integrator's repository through the SCM. Then, the pipeline updates all the software libraries and packages from the pre-configs in its server with a software package manager. For the Build FMU step, we preferred to use libraries and modules from the pre-configs, belonging to the open-source FMI Kit for Simulink, the commercial software tools Simpack and MATLAB without GUI, which helps us to overcome the OS dependency of the FMUs in the platform. In the next step, the source codes of the asset (incl. the simulation code script, use case specific code generator and some of the pre-configs) are analyzed and right after that their analysis reports are published in the code analysis tool. By using the FMPy Python package (see *GitHub - FMPy* (2024)), the two FMUs are easily validated and then their

metadata are directly extracted from themselves. After the pipeline updates the example SSP with these FMUs through Linux command executions, it directly executes the simulation code script to simulate the SSP as a black-box. For simulation purposes, we once again employ FMPy again and further developed the simulation code script, belonging to the SSP subfolder of the FMPy, for input parameterization and output generation. During the simulation testing progress, first, the input JSON, simulation results from SIMAT, and the SSP, containing the SSD system structure, SSV parameter files and FMUs, are read by the code script, then the simulation runs with a fixed step size and solver. Finally, the CSV results are generated for data analysis through the execution of the use case specific result generator code script, which generates validation curves and output JSON based on R4F standards. Penultimately, the Asset Integrator receives an email from the pipeline with its link, where they check all the console outputs, code analysis and simulation results. If satisfied with the entire asset integration and processing test, they deliver all asset files, including the SSP simulation results, and pipeline console outputs (excluding the pipeline code due to IP protection), to the version control system repository of the R4F Platform infrastructure layer.

6 Results and Discussion

In this section, first, we shortly discuss the simulation results coming from the FMI-based SSP simulation of the anti-slip traction & vehicle speed control co-simulation use case executed by the pipeline. Then, we address difficulties and challenges we faced related to the license management, pipeline configuration, automation and management of the asset integration and processing task in the pipeline on the R4F Platform.

6.1 Simulation Results

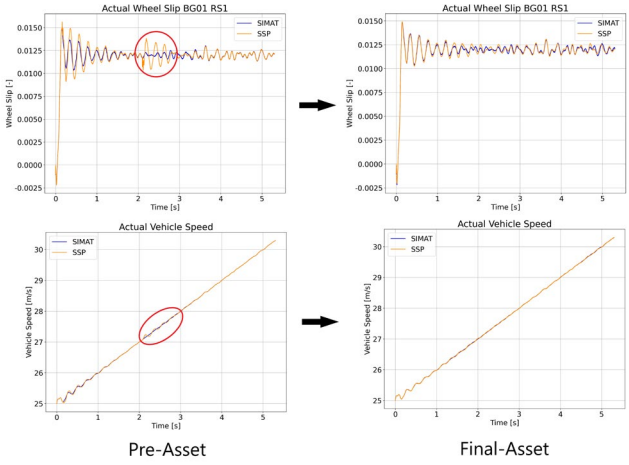


Figure 5. Result Validation and Optimization of the Anti-Slip Use Case in the CI/CD Pipeline.

In the left side of Figure 5, two different SIMAT and SSP simulation results of the pre-asset, belonging to the anti-slip use case, are displayed. These are actual values of the vehicle speed and longitudinal wheel slip of the first wheel set of the vehicle, which characterize the anti-slip use case. In the right side, the new SSP simulation results, coming from the final asset, are compared with the same SIMAT outputs after a demonstrative optimization of the SSP results by decreasing the simulation step size by a factor of ten, as previously discovered for this work. By this optimization, we aimed to reduce the unexpectedly arising little oscillations, differing from the SIMAT results and addressed with red circles in the figure. Besides that, the slip outputs give comprehensive insights about the anti-slip behavior of the model, which actually works well in the CI/CD pipeline, as realized in the figure. Lastly, it is remarkable that the constant acceleration of the vehicle functions in a right manner according to the relatively linear increase of the vehicle speed outputs.

To generate the whole picture in Figure 5, we used matplotlib, which is a Python library for visualization (see (Hunter 2007)). In general, the SSP simulation in the pipeline shows relatively consistent outputs with the SIMAT results by successfully optimizing the simulation. This also proves the success of our work with the pipeline at the end.

6.2 Challenges and Limitations

Software License Management: As mentioned in Section 1, we need software licenses, which are commercial and allow us to run the asset simulations after modeling the assets in their software. In the use case example, we needed the Simpack license to simulate the MBS model, and the MATLAB license to simulate the PID model for the SIMAT co-simulation. For the SSP co-simulation in the pipeline, we needed only the Simpack license and had to connect our VM to the Simpack license server through the use of the Virtual Private Network service by logging in with our username and password. Moreover, we used these licenses to automatically build the FMUs of these models in the pipeline without opening any GUI window. In addition, we had a Model.CONNECT license, by which we built the SSP example file of the complete anti-slip co-simulation model once as mentioned in Section 5.

Pipeline Installation and Configuration: To implement the pipeline in a right manner, we as Asset Integrator needed much know-how and experience with DevOps practices regarding to the CI/CD pipeline technology. Especially, it was very important to get to know how to install and configure the VM, software tools, packages, libraries, and then interoperate them with each other in harmony. For this work, we decided to use plugins in the pipeline server for the interoperation, code analysis tool for code analysis, version control system for asset tracking and storing, CLI for pipeline remote control, mailing, asset pushing, and basic execution commands, belonging to the software tools, packages and libraries, to apply the sub-processes defined in Subsection 4.4 to the pipeline.

Automation Testing: After providing the necessary soft-

ware licenses, installing and configuring our pipeline, we needed to test the entire asset integration and processing incl. the asset simulation as automated in the pipeline. Therefore, we always needed to do research, find out possible ways and then try them step by step to make every sub-process, implemented in the pipeline code, working in a right manner. Besides, there are use case specific limitations for the simulation test in the pipeline. For example, the anti-slip use case shows smaller oscillations by refining the simulation step size, which significantly improves the quality of the simulation results while increasing the simulation runtime on the contrary as previously discovered in this work.

7 Conclusion and Outlook

Based on our experience, the CI/CD pipeline technology is practical and interactive for automating and managing the entire FMI- and SSP-based asset integration and processing with the version control system and code analysis tool in a collaborative work environment on the R4F Platform. In addition, the FMI and SSP standards greatly facilitated the adaptive simulation of the assets in the platform, specifically in terms of tool independence, asset description and file portability. This was also demonstrated in the work of (Kugu et al. 2023). Furthermore, we successfully co-simulated multiple FMUs as one SSP for the anti-slip use case, which has shown relatively consistent results based on the validation curves in Figure 5. For this succession, we encountered challenges and restrictions related to the license management, pipeline configuration, automation and management of the asset integration and processing task in the pipeline on the platform, which should not be neglected.

In future, we plan to fully automate the asset integration and processing task by automating the model preparation and validation processes in the pipeline. In addition, we consider combining the automation and management approach with the dynamical and auto-pipeline generation (see (Reiterer, Schiffer, and Benedikt 2022)) and visualization prototype to enhance the DevOps practice in the platform. Furthermore, we plan to find out solutions to completely ensure the IP protection of the simulation models in the platform. We plan to simulate these models as servers connected to the pipeline without uploading them into the platform.

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

Barbie, Alexander, Wilhelm Hasselbring, and Malte Hansen (2023). "Enabling Automated Integration Testing of Smart Farming Applications via Digital Twin Prototypes". In: *2023 IEEE Smart World Congress (SWC)*. IEEE, pp. 1–8. DOI: 10.1109/SWC57546.2023.10449240.

Functional Mock-up Interface (FMI) Standard (2024). URL: https://fmi-standard.org/ (visited on 2024-08-09).

Git Documentation (2024). URL: https://git-scm.com/docs/git (visited on 2024-04-18).

GitHub - FMI Kit for Simulink (2024). URL: https://github.com/CATIA-Systems/FMIKit-Simulink (visited on 2024-04-18).

GitHub - FMPy (2024). URL: https://github.com/CATIA-Systems/FMPy (visited on 2024-04-18).

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. DOI: 10.1177/09544097211039395.

Hällqvist, Robert et al. (2021). "Engineering domain interoperability using the system structure and parameterization (ssp) standard". In: *Proceedings of 14th Modelica Conference 2021, Linköping, Sweden, September 20-24, 2021*, pp. 37–48. DOI: 10.3384/ecp2118137.

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. DOI: 10.1038/s41598-022-18452-w.

Hotzel Escardo, Tomas et al. (2021). "Modelling train driver behaviour in railway co-simulations". In: *Software Engineering and Formal Methods. SEFM 2020 Collocated Workshops: ASYDE, CIFMA, and CoSim-CPS, Amsterdam, The Netherlands, September 14–15, 2020, Revised Selected Papers 18*. Springer, pp. 249–262. DOI: 10.1007/978-3-030-67220-1_19.

Hugues, Jerome et al. (2020). "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*. New York, NY, USA: Association for Computing Machinery, pp. 1–5. DOI: 10.1145/3417990.3421446.

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, Simon (1998). "The Manchester Benchmarks for Rail Vehicle Simulation". In: *Vehicle System Dynamics* 30.3-4, pp. 295–313. DOI: 10.1080/00423119808969454. URL: https://api.semanticscholar.org/CorpusID:110412927.

Kaewunruen, Sakdirat et al. (2023). "Digital Twins for Managing Railway Bridge Maintenance, Resilience, and Climate Change Adaptation". In: *Sensors* 23.1. ISSN: 1424-8220. DOI: 10.3390/s23010252. URL: https://www.mdpi.com/1424-8220/23/1/252.

Kiran, Kumar H K et al. (2021). "An Approach to basic GUI-enabled CI/CD pipeline with Static Analysis tool". In: vol. 23. 6. Journal of University of Shanghai for Science and Technology, pp. 683–693. DOI: 10.51201/JUSST/21/05317.

Kugu, Ozan et al. (2023). "An FMI-and SSP-based Model Integration Methodology for a Digital Twin Platform of a Holistic Railway Infrastructure System". In: *Proceedings of the 15th International Modelica Conference 2023, Aachen, October 9-11*, pp. 717–726. DOI: 10.3384/ecp204717.

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. DOI: 10.1109/MECO.2018.8406023.

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. DOI: 10.3390/electronics11203325.

Reiterer, Stefan H, Clemens Schiffer, and Mario Schwaiger (2023). "A graph-based meta-data model for devops: Extensions to ssp and sysml2 and a review on the dcp standard". In: *Proceedings of the 15th International Modelica Conference 2023, Aachen, October 9-11*, pp. 159–166. DOI: 10.3384/ecp204159.

Sethi, Farhana (2020). "Automating software code deployment using continuous integration and continuous delivery pipeline for business intelligence solutions". In: *Authorea Preprints*. DOI: 10.22541/au.160373745.57814465/v1.

System Structure and Parameterization (SSP) Standard (2024). URL: https://ssp-standard.org/ (visited on 2024-08-09).

Villa, Davide et al. (2024). "Colosseum as a digital twin: Bridging real-world experimentation and wireless network emulation". In: *IEEE Transactions on Mobile Computing*, pp. 1–17. DOI: 10.1109/TMC.2024.3359596.

Zampetti, Fiorella, Simone Scalabrino, et al. (2017). "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)*. IEEE, pp. 334–344. DOI: 10.1109/MSR.2017.2.

Zampetti, Fiorella, Damian Tamburri, et al. (2023). "Continuous integration and delivery practices for cyber-physical systems: An interview-based study". In: *ACM Transactions on Software Engineering and Methodology* 32.3, pp. 1–44. DOI: 10.1145/3571854.

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)*, pp. 430–433. DOI: 10.1109/DTPI52967.2021.9540118.

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. DOI: 10.1016/j.procir.2022.05.263.

Zhou, Shiyang, Ozan Kugu, et al. (2024). "A Reinforcement-Learning-based Parameter Tuning Methodology for Traction Control in the Holistic Railway Digital Twin System". In: *Procedia CIRP* 128, pp. 828–833. DOI: 10.1016/j.procir.2024.06.040.

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. DOI: 10.1016/j.procir.2023.02.141.