Using SUMO for Test Automation and Demonstration of Digitalized Railway Concepts

Arne Boockmeyer, Dirk Friedenberger, and Lukas Pirl

Operating Systems and Middleware Chair,

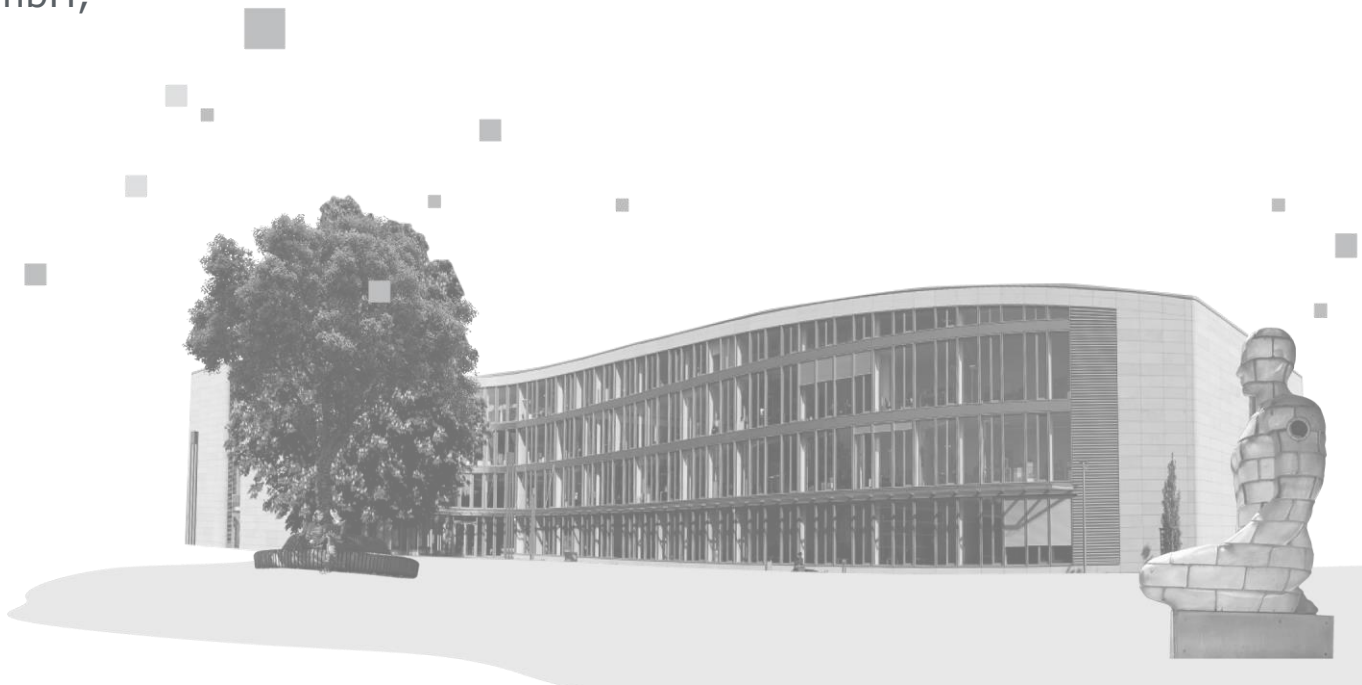
Prof. Dr. Andreas Polze,

Hasso Plattner Institute for Digital Engineering gGmbH,

University of Potsdam

HPI Hasso Plattner Institut

Digital Engineering · Universität Potsdam



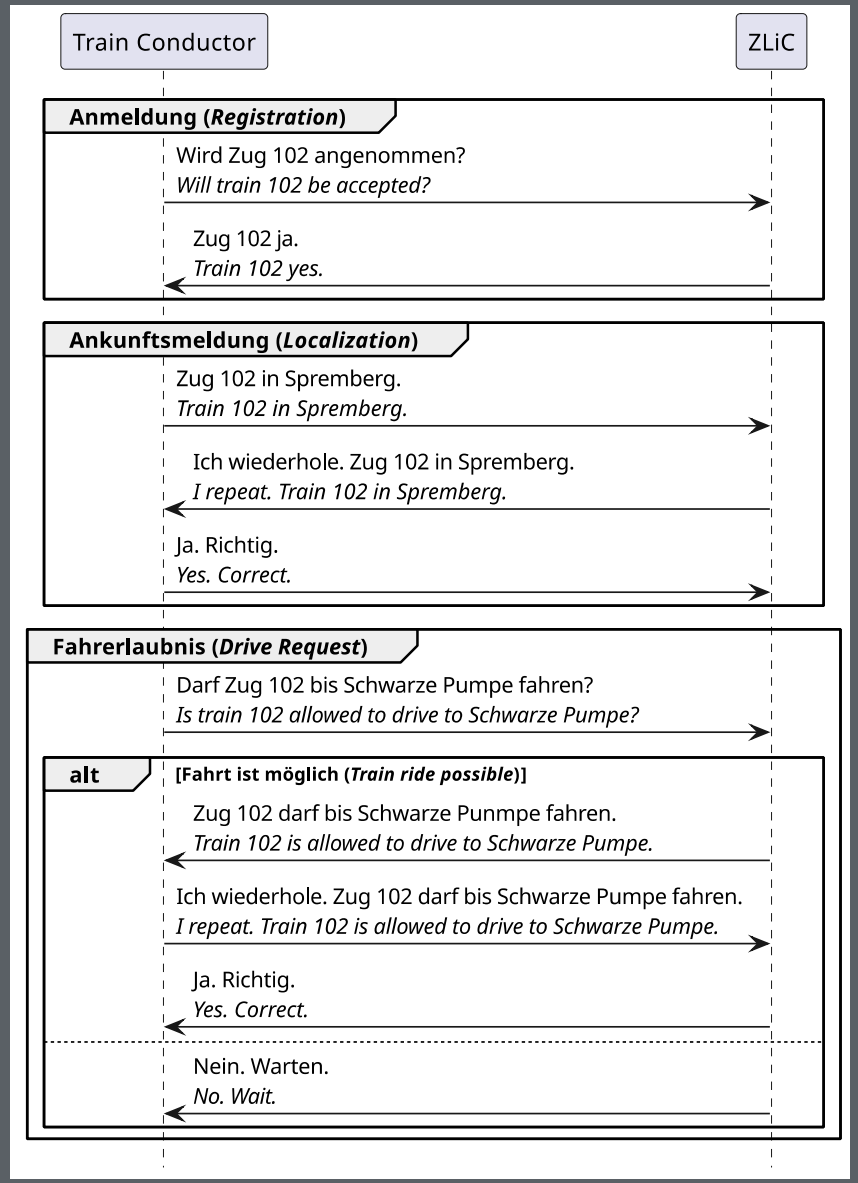
Design IT.
Create Knowledge.

www.hpi.de

Introduction

- Due to the climate change, there is a shift in the energy industry, affecting coal mining regions, such as the Lusatia region in southern Brandenburg
- The biggest employers of that region facing challenges during the next years and it’s unclear, how the region and its population develop the next years
- The goal of the FlexiDug project, funded by the Ministry for Digital and Transport, is to develop a lightweight railway operation procedure to enable public transport and freight transport in that region by reusing the existing infrastructure
- One result of this project is the Train Dispatcher in the Cloud (ZLiC)

- This talk (and the paper) is about using SUMO as part of the ZLiC, to support the development by having an opportunity for **visualization** and **test automation**



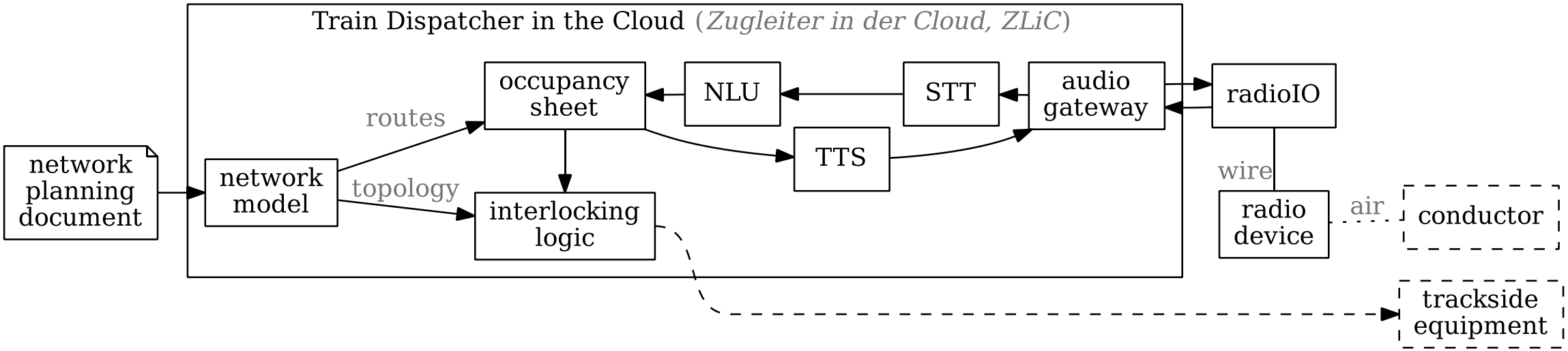
Train Dispatcher in the Cloud (ZLiC)

- Cloud-based approach to digitalize the German *Zugleitbetrieb* (comparable to American Track Warrant Control)
- Lightweight railway operating procedure, following a voice-based protocol between the train conductor and ZLiC
- Goal is to replace the human train dispatcher but keeping the same interface
- It supports the three major operation in the *Zugleitbetrieb* according to Ril 436: Registration, Localization and Driving Requests

L. Pirl, H. Herholz, D. Friedenberger, A. Boockmeyer, A. Polze, and B. Milius, "Train dispatcher in the cloud — digitalizing track warrant control for safe train operations in structurally transforming areas," Transport Research Arena 2024, 2024.

ZLiC System Architecture

The system covers multiple components, communicating via a data distribution framework:



- The ZLiC receives the voice commands via the audio gateway
- Speech to text and natural language understanding frameworks transform the input to commands
- The logic side, an occupancy sheet and an interlocking logic initialized with a network model, processes the commands
- A text to speech library produces the output for the train conductor

Use Cases for Simulations

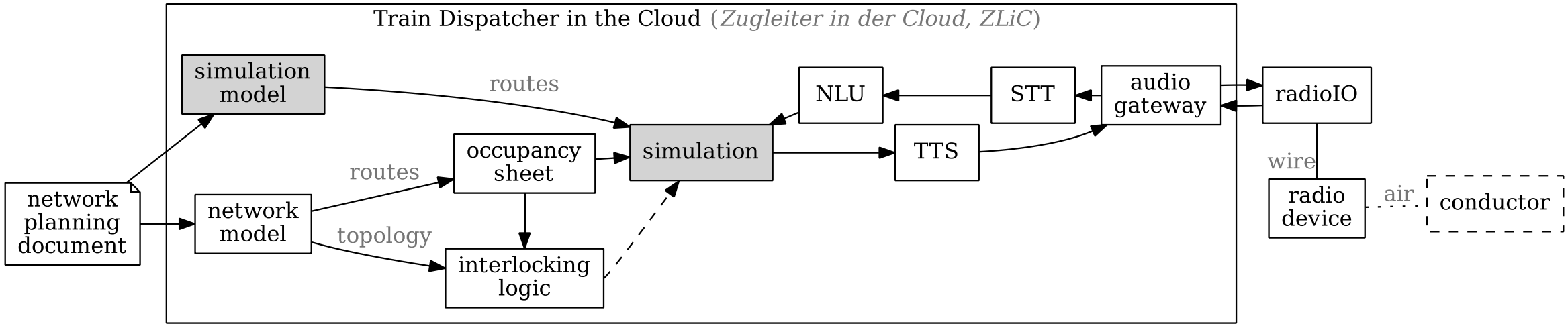
As part of the development of the ZLiC we integrated SUMO for two purposes:

Visualization	Test Automation
• Performing dry runs (without real trains and railway infrastructure) lacks the possibility of visual “moving” output • Important for understanding the system • SUMO adds moving traffic besides the graphical occupancy sheet and the voice communication	• The voice-based interface makes it intricate to run tests, since its laborious to create the input • Large experiments are hard to realize • A traffic simulation environment increases test realism especially regarding the behavior of the trains • The SUMO component can inject automated trains to the simulation to add additional traffic

- Additional use cases to use SUMO are possible but not in the focus right now
- Ideas are getting deeper in dispatching of trains or timing predictions using simulations

Injection of SUMO as Simulation Environment

- The ZLiC system is developed following a *ontology-based system* approach
- Injecting a SUMO component required just a new component and the rerouting of the messages:



- The simulation component is a docker container, containing a control logic, SUMO, and TraCI
- The interlocking logic communicates with the simulation to control the SUMO signals

Integration of SUMO

To integrate SUMO, we had to do three steps:

1
- Creation of the SUMO simulation model, containing the network model and routes
- Using the same data source as the logic components of the ZLiC

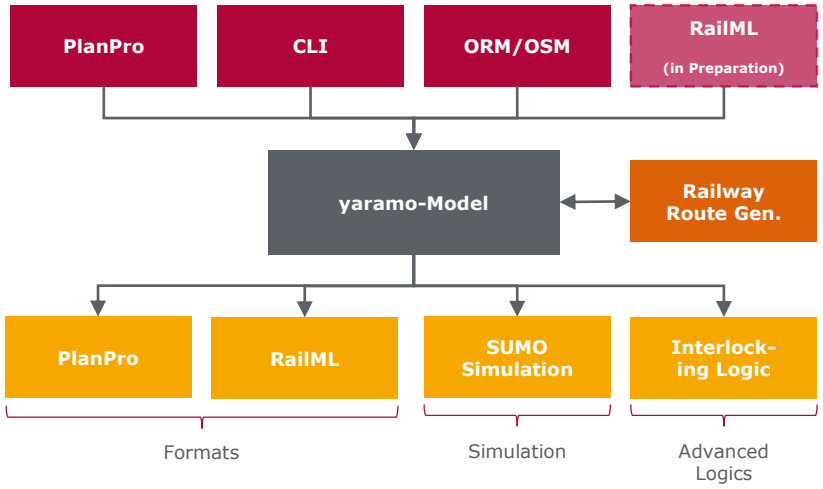
2
- Capture the commands between the NLU and the occupancy sheet
- Apply the same commands to the SUMO simulation

3
- Implement an option for automated trains
- Communicate in all three directions: To the occupancy sheet, to the TTS, and SUMO

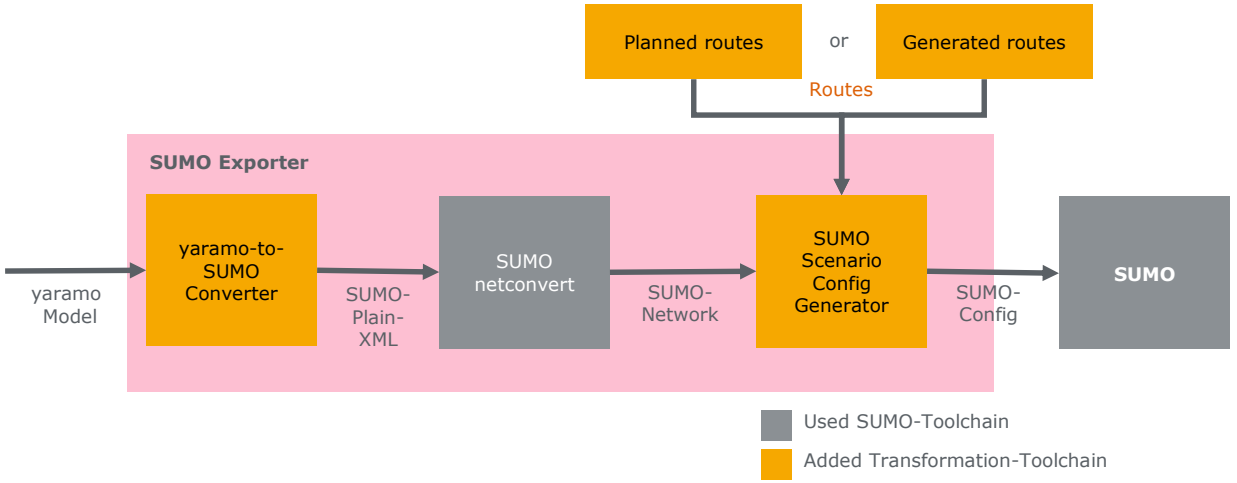
Creation of the SUMO Simulation Model

- To create the SUMO simulation model, containing the network model and routes through the network, we are using the **yaramo** toolchain
- Input are either digital railway planning documents or open data sources such as ORM/OSM
- A SUMO exporter creates the SUMO simulation model, but the same yaramo instances is used as parameter for the interlocking logic and the digital occupancy sheet

The yaramo toolchain:

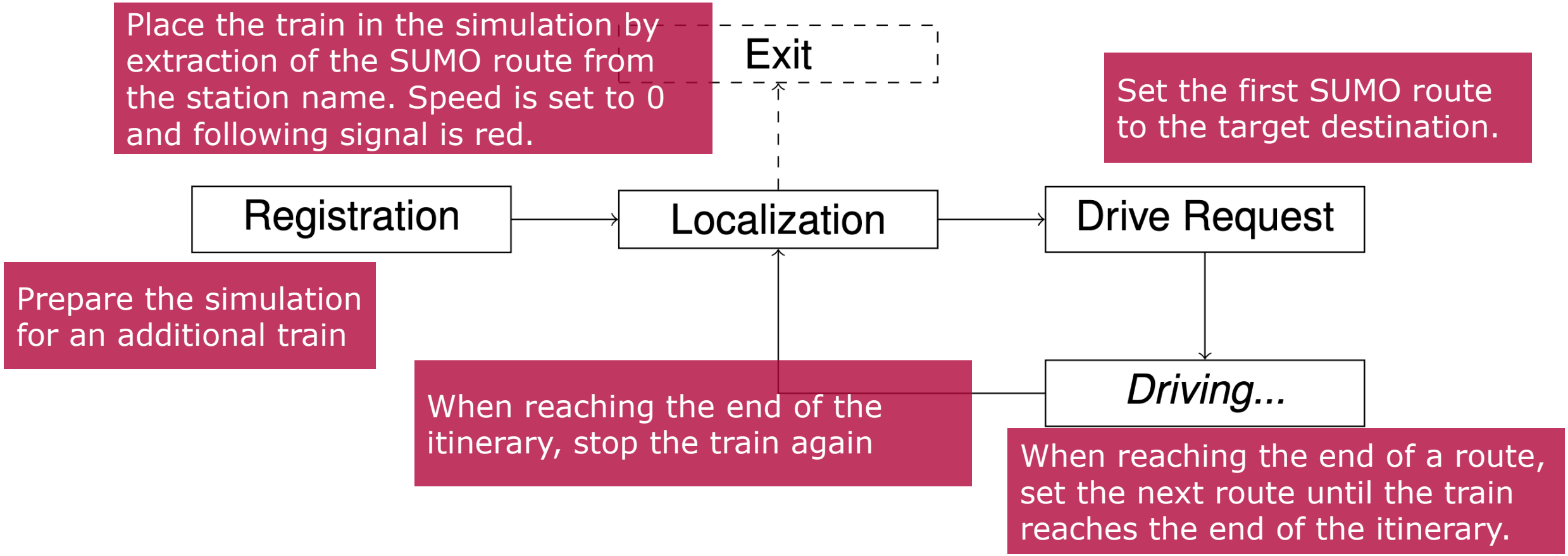


The SUMO exporter:



Capture the Commands between the NLU and OS

- Three separate commands needs to be extracted between the NLU and the occupancy sheet
- Fulfils requirements for the first use case



3

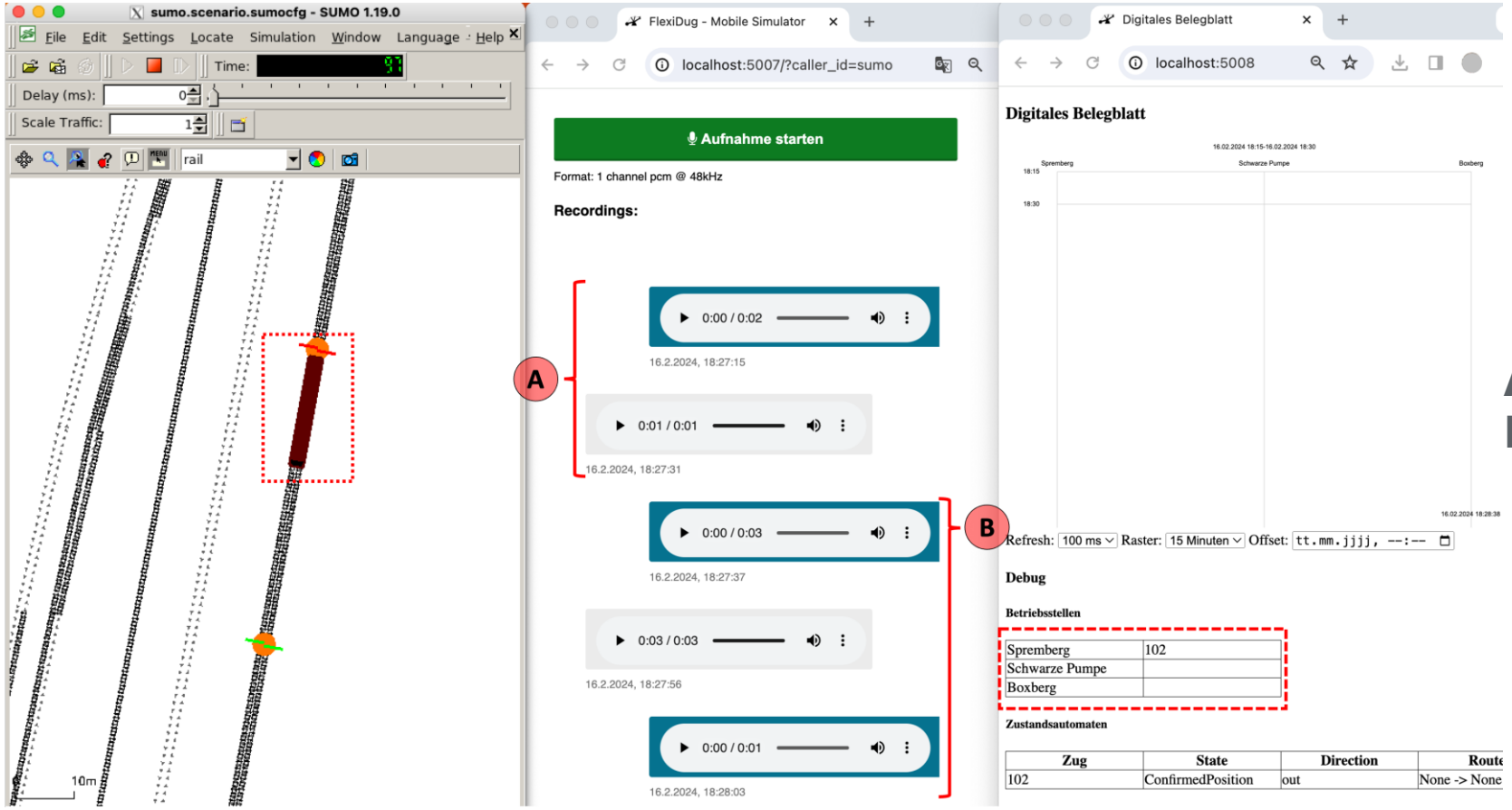
Add Additional Automated Trains

- Automated trains are necessary for the second use case: Test Automation
- Enables large experiments, with several trains
- Automated trains are configured inside of the code (so far):

```
self.automatic_trains.append(AutomaticTrain( train_id: "203",  create_after: 10,  start_position: "Spremberg",
                                              end_position: "Boxberg", intermediate_stops=["Schwarze Pumpe"]))
```

- Uses the same interfaces as the “manually spoken” traffic
- Produces the voice output of the train conductor as well
- Requires additional data exchange with SUMO: When reaching the end of an itinerary (detected by the edges IDs), the automated train triggers the localization message

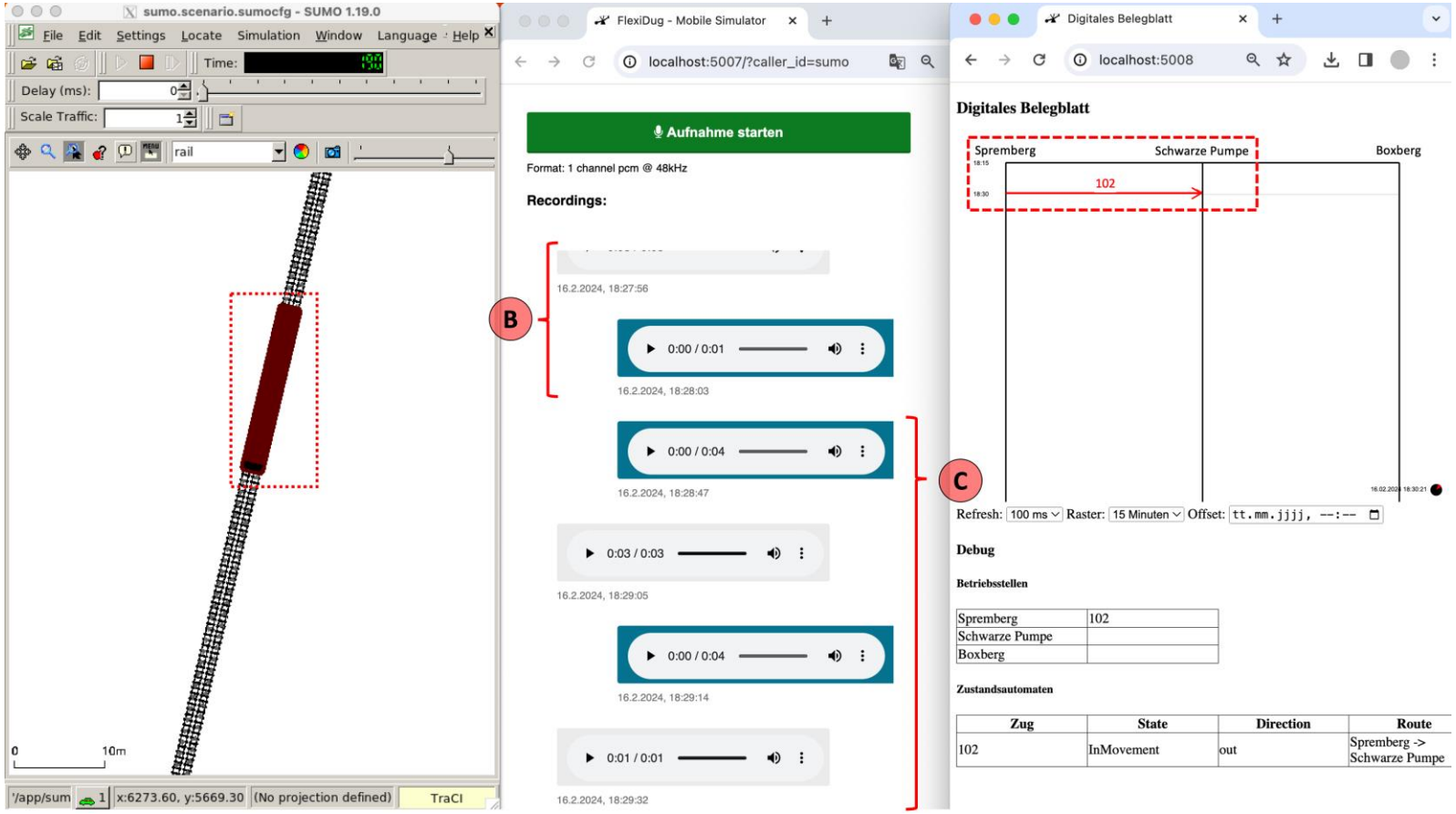
Experiment 1: Voice-based Interface and Visualization



SUMO **Voice-Interface** **Occupancy Sheet**

A: Registration
B: Localization

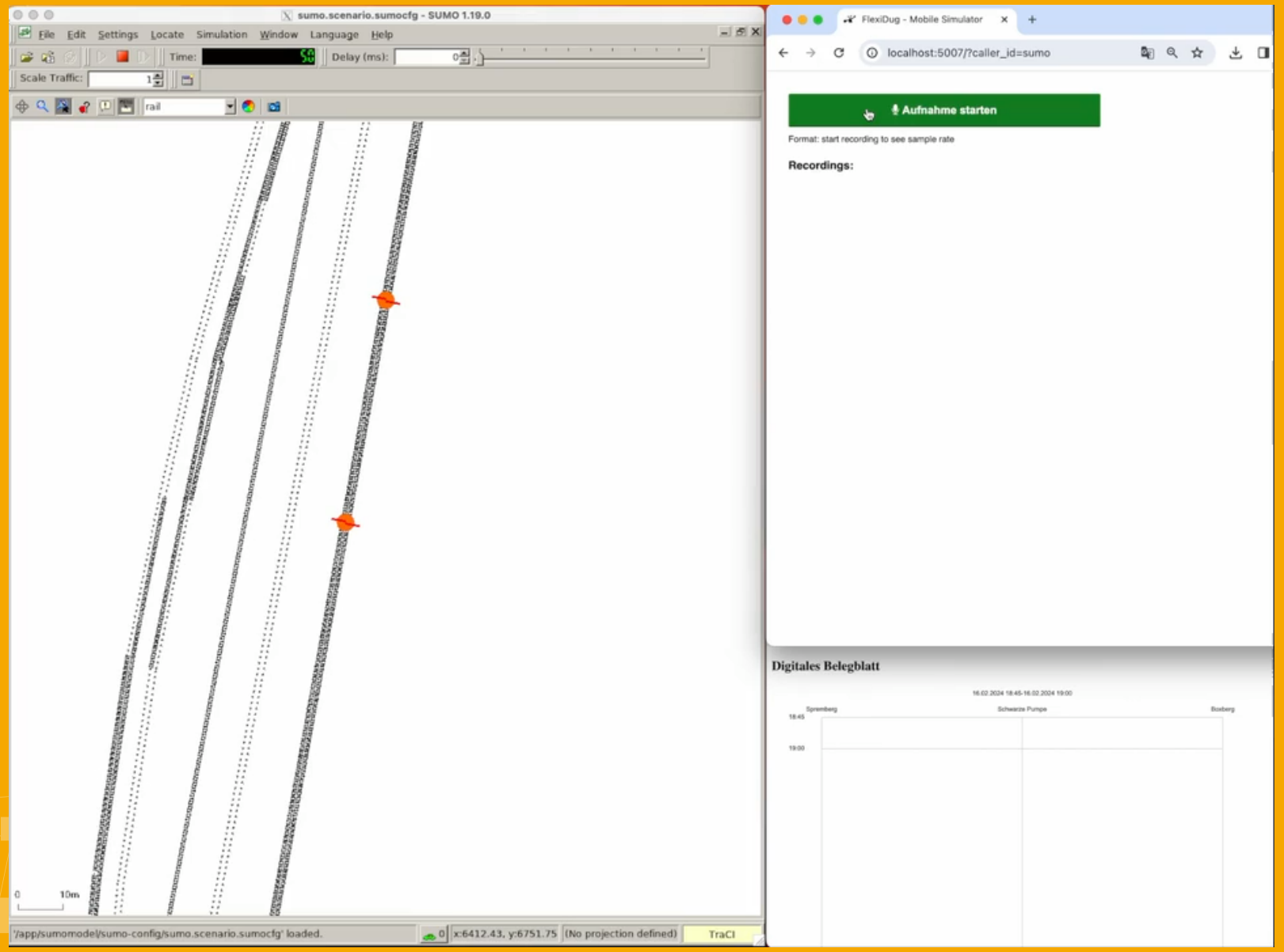
Experiment 1: Voice-based Interface and Visualization



B: Localization
C: Driving Request

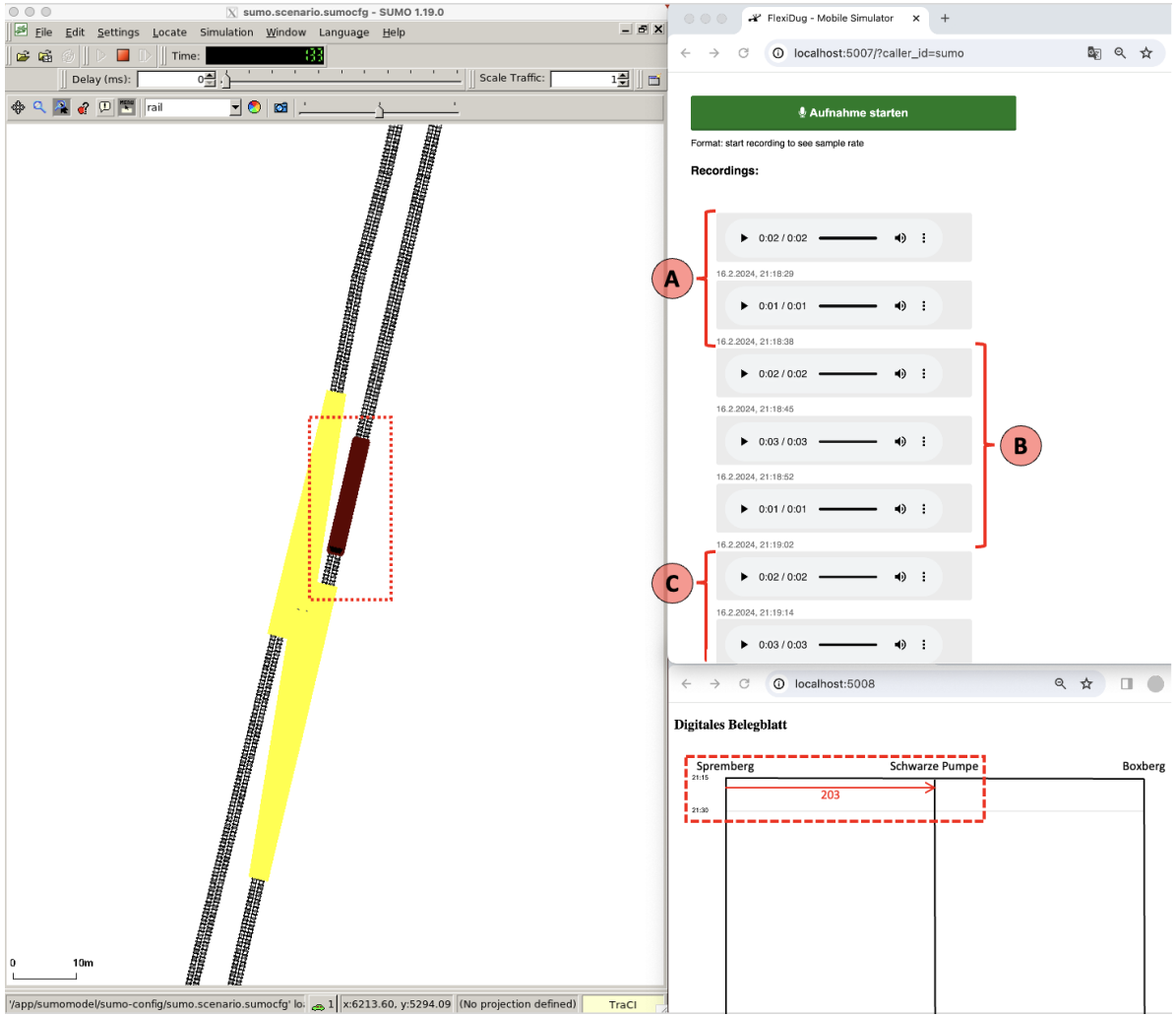
SUMO **Voice-Interface** **Occupancy Sheet**

Video 1:



Video: https://osm.hpi.de/staging/flexidug/sumo-user-conference-2024/video-1

Experiment 2: Automatic Train Operations

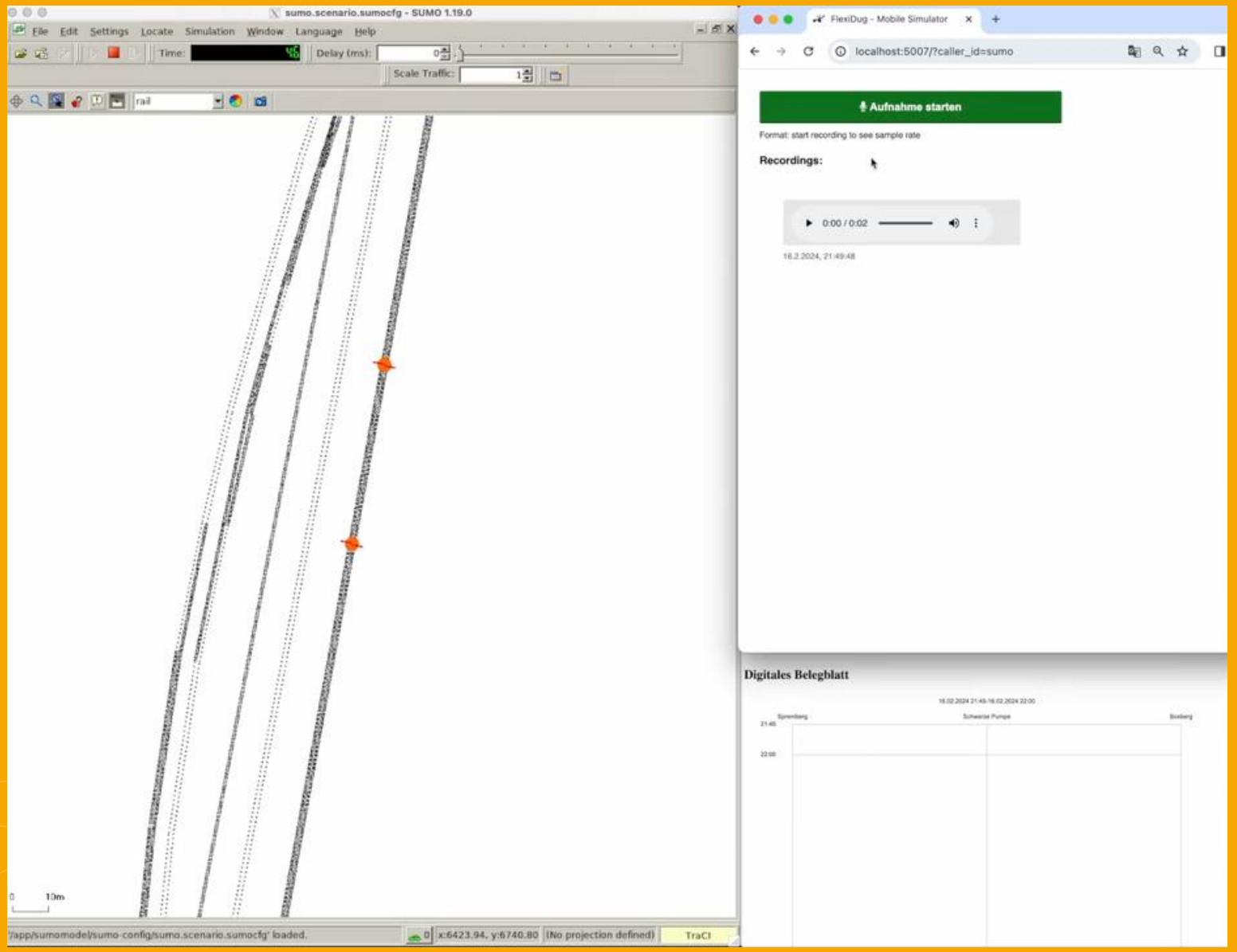


A: Registration
B: Localization
C: Driving Request

Configuration:

```
self.automatic_trains.append(
    AutomaticTrain( train_id: "203",
                    create_after: 45,
                    start_position: "Spremberg",
                    end_position: "Boxberg",
                    intermediate_stops=["Schwarze Pumpe"]))
```

Video 2:



Video: https://osm.hpi.de/staging/flexidug/sumo-user-conference-2024/video-2

Next Steps / Future Work

Test Suite Creation

- Complete implementation of test suite
- Test-case definition outside of the implementation
- Automated evaluation of parameters, such as arrival times, …

Include Voice Interface

- Move / Add simulation component before the audio gateway
- Tests would also include audio interface and processing
- Self-recorded audio files (e.g. with accents) would increase realism

Multi-Line Support

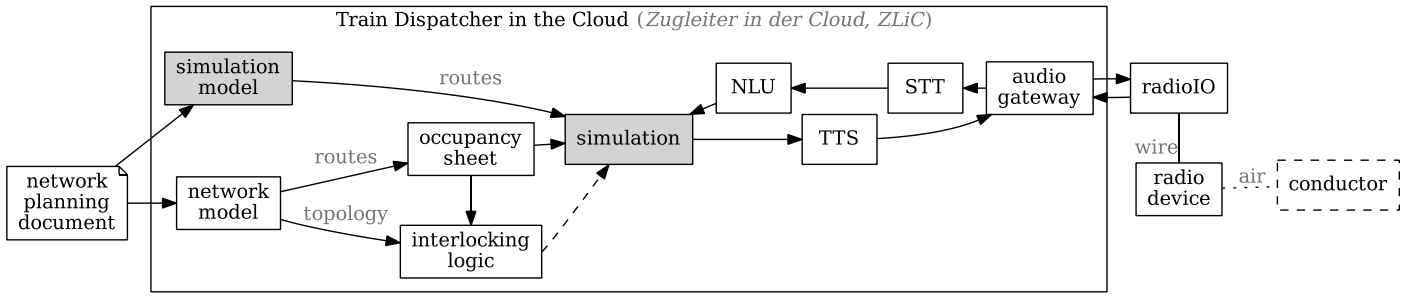
- ZLiC only support single lines now
- Additional lines would be more interesting for further applications in real-world
- Even interfering lines would be possible

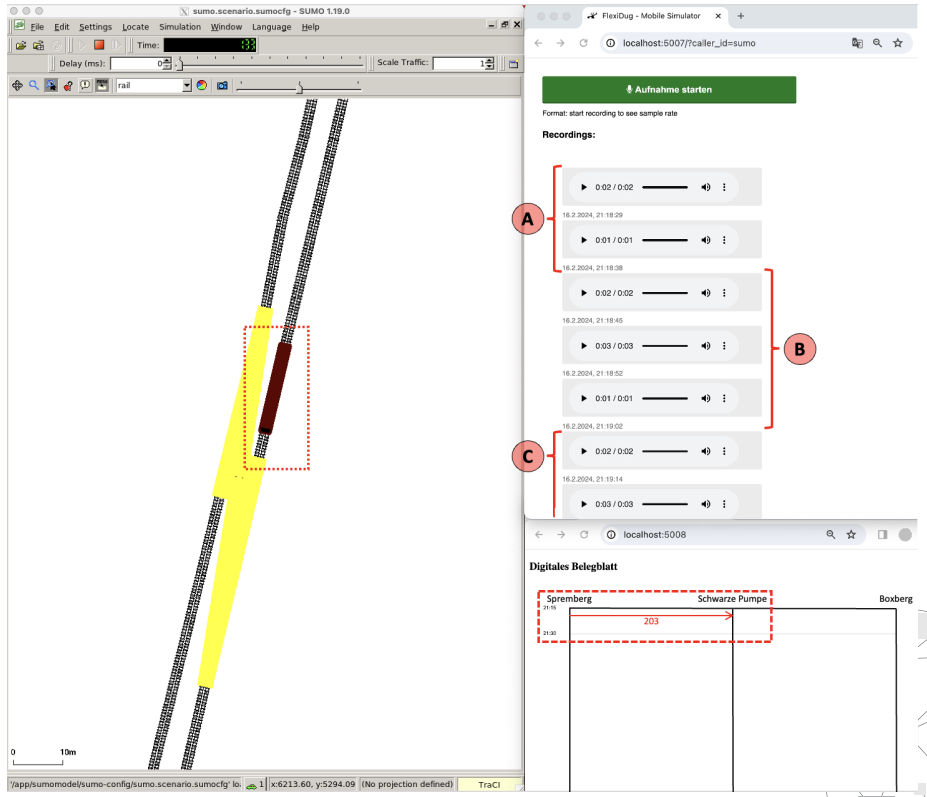
Use SUMO for Predictions

- Deeper look into dispatching algorithms could lead towards an enhanced ZLiC
- ZLiC could automatically react to unexpected incidents
- During decision making, SUMO could support with fast simuations

Using SUMO for Test Automation and Demonstration of Digitalized Railway Concepts

Arne Boockmeyer, Dirk Friedenberger, and Lukas Pirl





Design IT.
Create Knowledge.

www.hpi.de

Backup-Slides

From: A. Boockmeyer, J. Baumann, B. Schenkel, et al., “Processing digital railway planning documents for early-stage simulations of railway networks,” Transport Research Arena 2024, to appear, 2024.

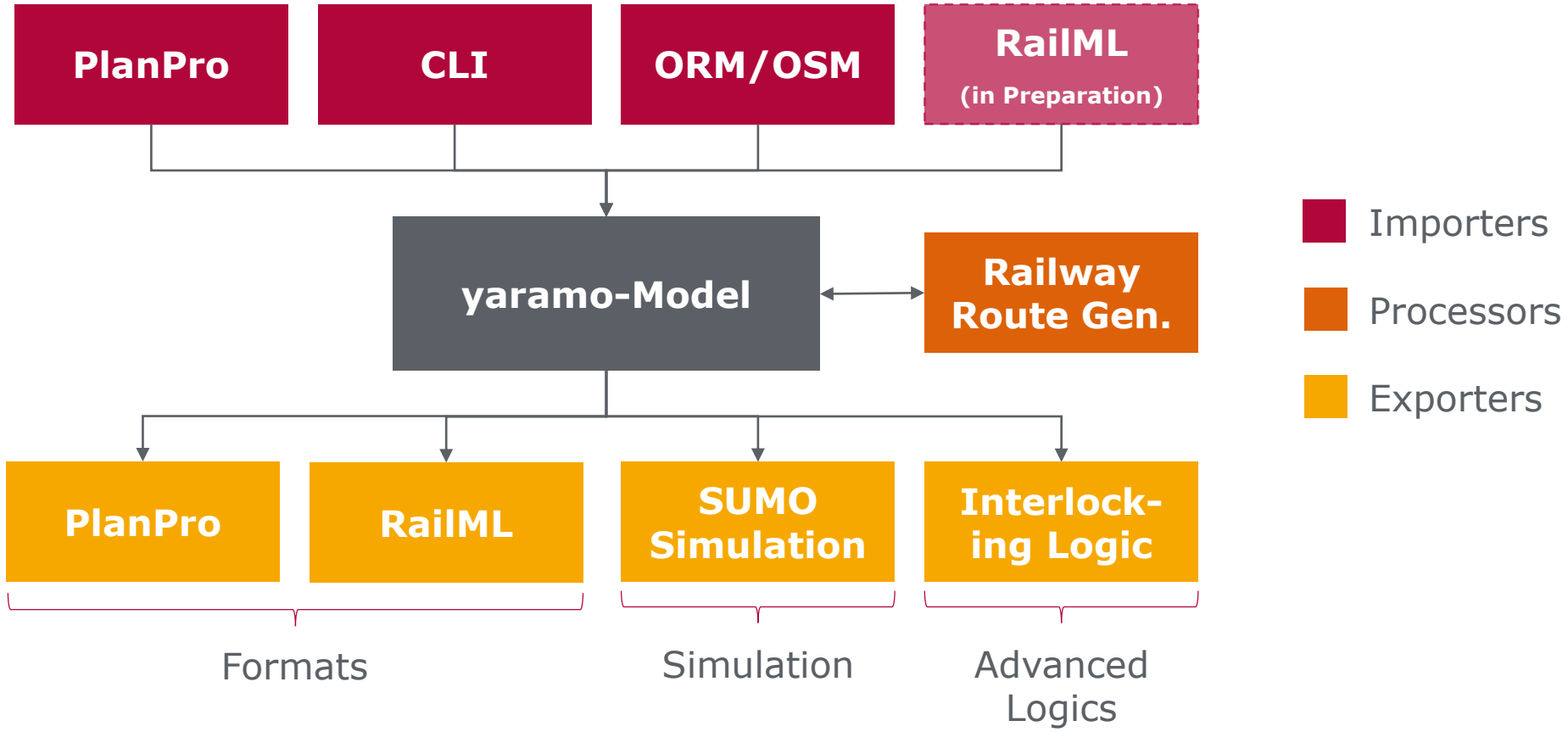
Railway Planning Documents

- Railway Infrastructure Provider and Industry are moving away from paper-based plans towards fully digitalized planning processes.
- The digital documents contain details about:
  - The Topology (points, rails, …)
  - The Geography (location of tracks, elements, …)
  - The CCS infrastructure (signals, train detection systems, …)
- For us, two formats / data sources are mostly relevant:

PlanPro	Open Railway Map / Open Street Map
• Developed by Deutsche Bahn started in 2009 • XML-based format, covering all details required for an ESTW • "Werkzeugkoffer" as tool for visualization and analysis	• Publicly available data sources collected by volunteers • Similar data structure, but quality depends on the effort during the data collection • Largely available all over the world

yaramo Model / Toolchain / Architecture

Shared models containing the topology and geography of railway networks



Use Cases of Digital Railway Planning Documents

- Besides the advantages during the planning process, digital planning documents can be used for several more applications:

Early-Stage Simulations of Railway Plannings

- Digital planning data (for example in the PlanPro format) can be converted to yaramo models.
- A SUMO exporter converts this model to a simulation models
- This enables railway simulations in early stages of the planning document

Testautomation of CCS Infrastructure

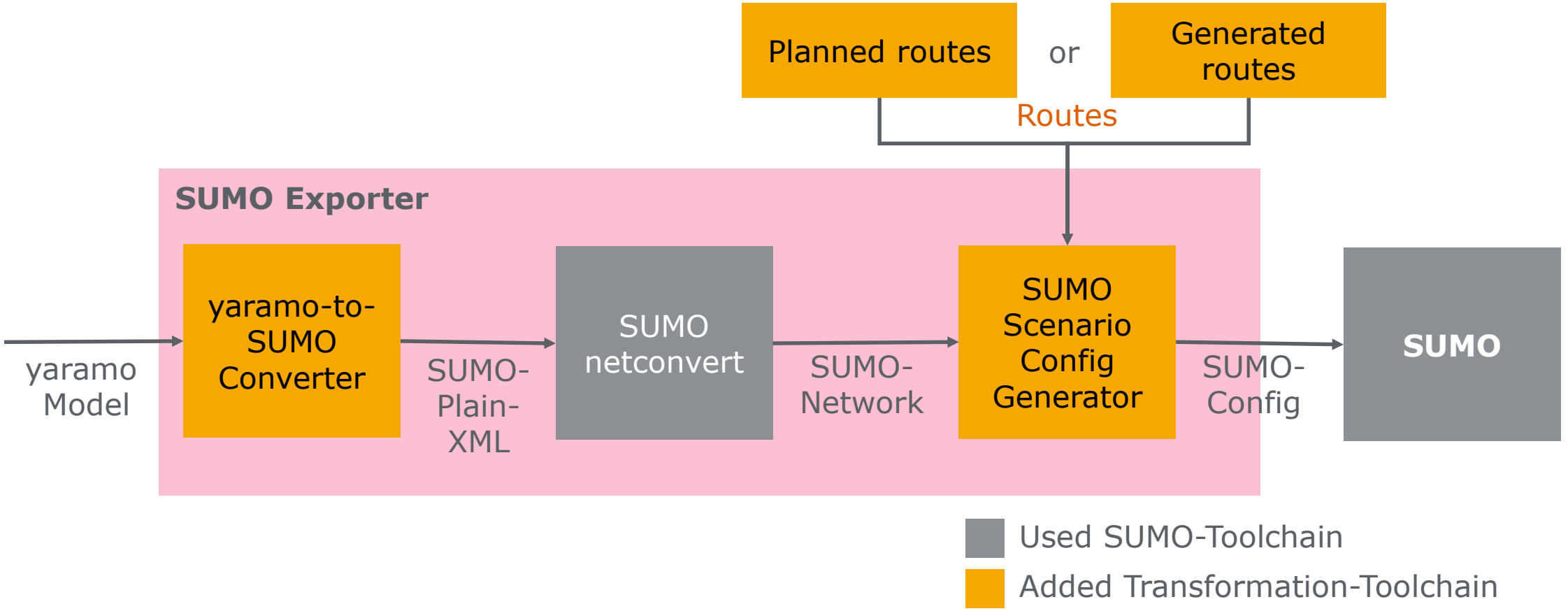
- With a generic interlocking logic, parameterized with a yaramo model, and the simulation, hardware elements, such as object controllers, can be tested automatically
- Simulation generates many operating procedures; interlocking logic controls the hardware elements

Support Railway Research Projects

- Using real railway networks as input and the simulation capabilities as driver for research projects
- Ideas are for example the generation of traffic as load for the projects or high-quality data inputs for railway networks

Conversion of yaramo Models to SUMO Simulation Models

- To transform a yaramo model to a SUMO simulation, several steps, including graph transformations, are necessary:



```
# Using PlanPro as Data Source
reader = PlanProReader("MVP")
topology =
     reader.read_topology_from_plan_pro_file()


# Processor
generator = RouteGenerator(topology)
generator.generate_routes()


# Export to a SUMO Simulation Model
sumo_exporter = SUMOExporter(topology)
sumo_exporter.convert()
sumo_exporter.write_output()
```

yaramo/SUMO Code Example

- Programming language is Python
- The example on the left shows:
  - A PlanPro file „MVP.ppxml“ as source will be extracted
  - This file contains the topology and geography, but routes need to be generated in the second step
  - At the end, the enriched data are extracted to a SUMO simulation model
- Publicly accessible on GitHub:

  https://github.com/simulate-digital-rail/