Simulation
Stakeholders
Domain Experts
Models
Communication
Management
Test and Verification

FlexiDug: Communication with the "Zugleiter in the Cloud"

Dirk Friedenberger, Arne Boockmeyer, Lukas Pirl

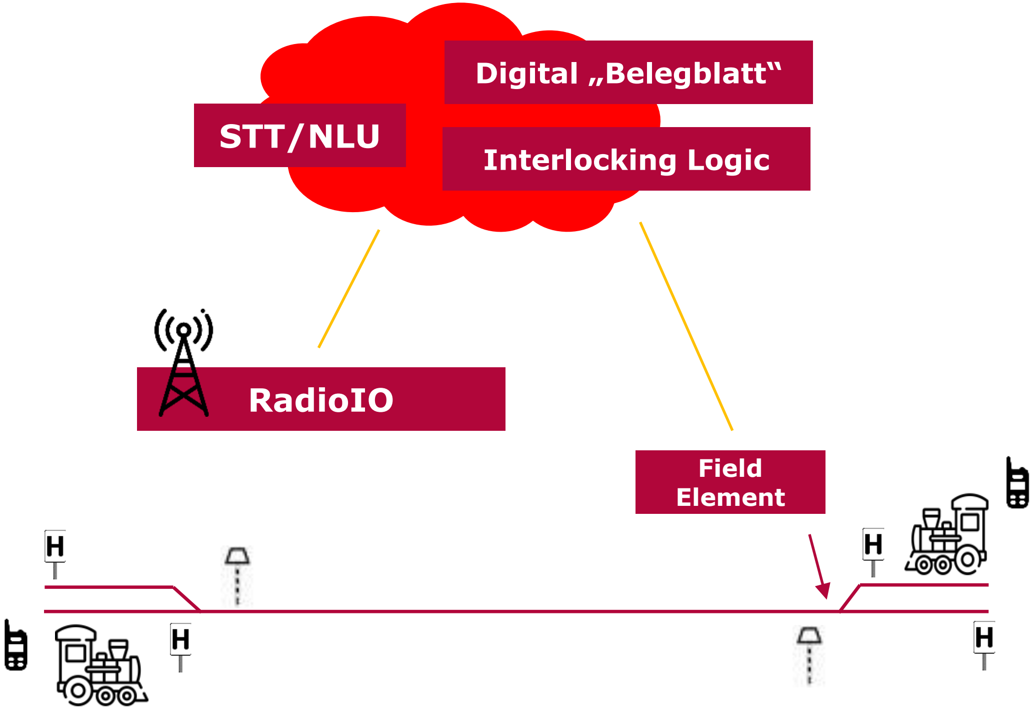
18.09.2023

HPI Hasso Plattner Institut
Digital Engineering • Universität Potsdam

FlexiDug

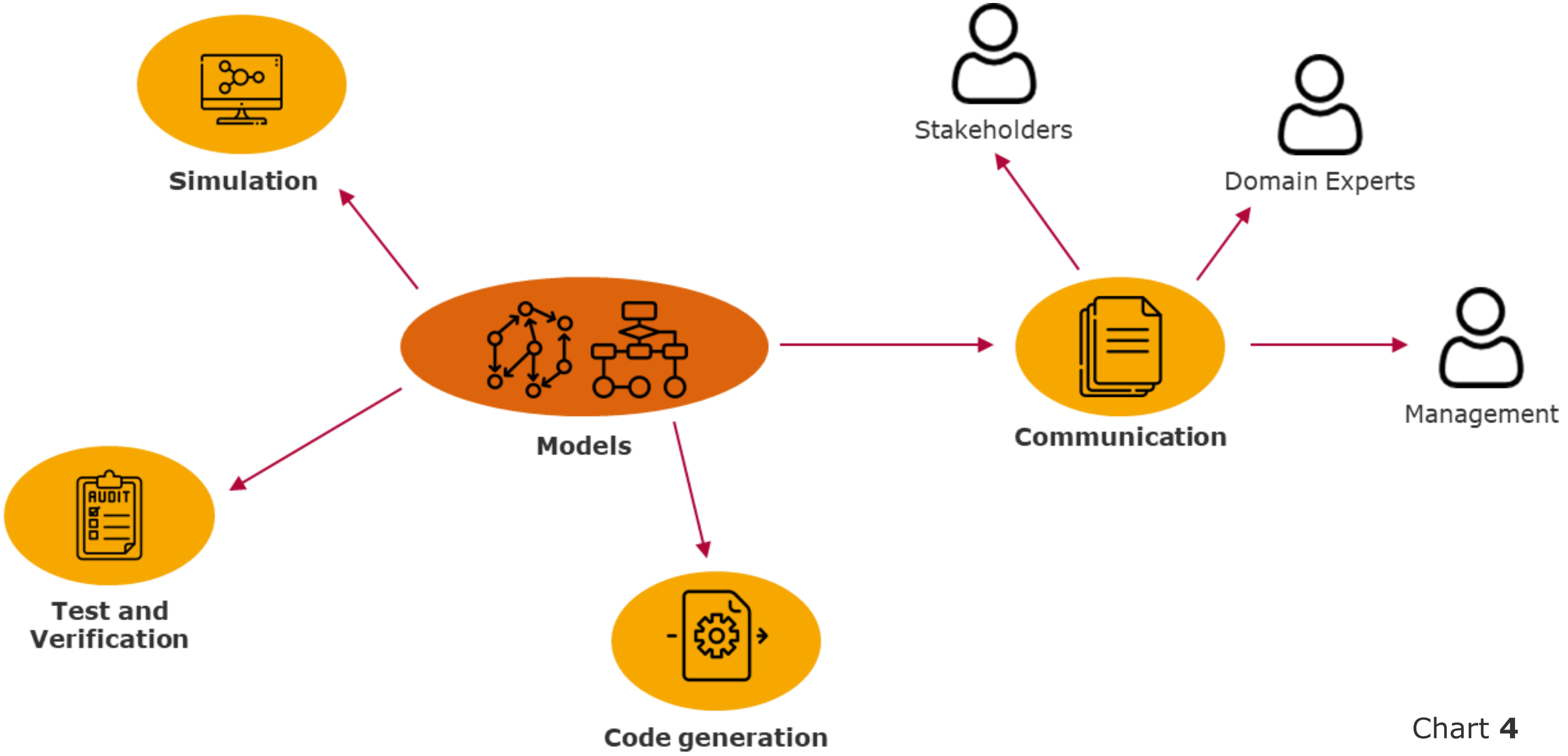
- Coal mining areas, such as Lusatia region, face structural transformations
- Digitalisation offers opportunities in growth regions
- Economic and flexible solutions for rail transport
- Reusing mining railway infrastructures
- FlexiDug has three technical focus areas: Digital Twin, Sensor Networks, Flexible Digital control, command and signalling (CCS)

Flexible Digital Control, Command and Signalling (CCS)

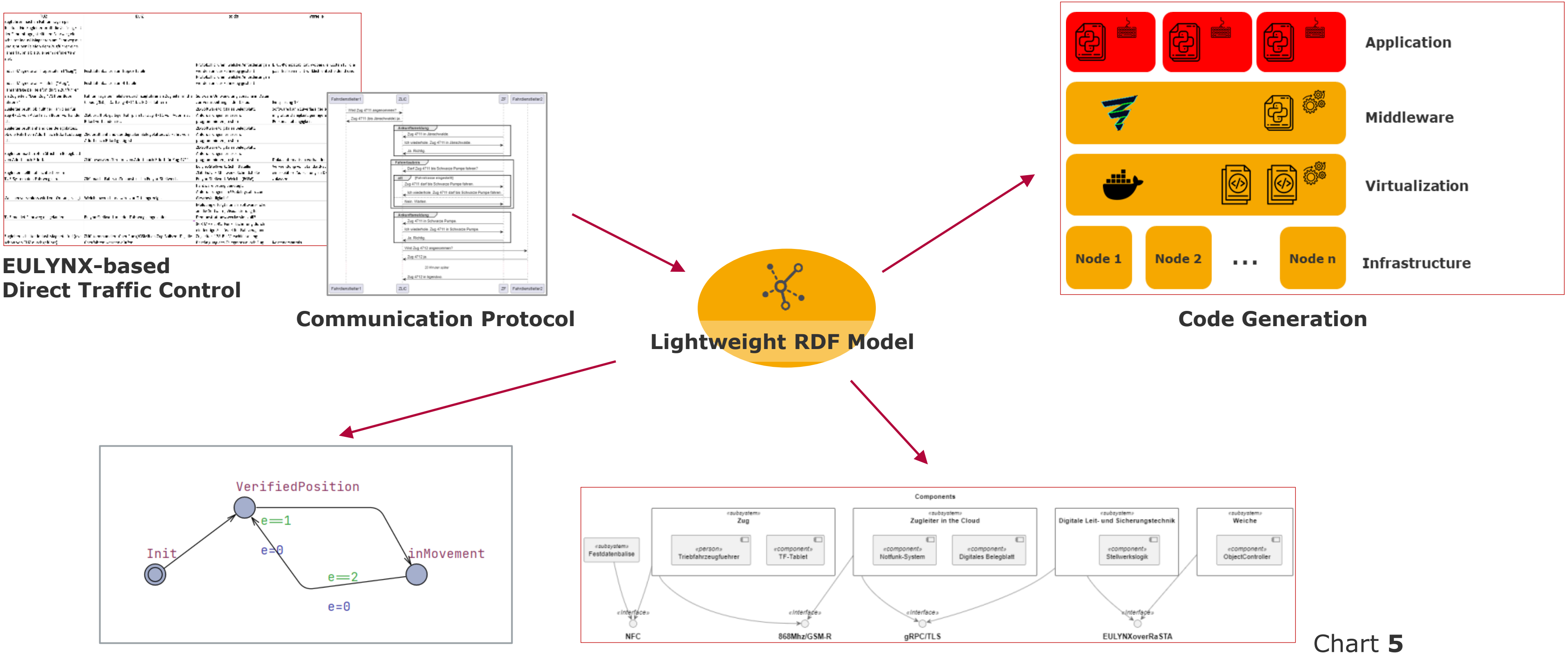


- Flexible
  - □ Freight transport ⇔ Passenger transport
  - □ Industrial railway / Narrow-gauge railway
- Digital
  - □ Hardware replaced by software
  - □ Digital version of German „Zugleitbetrieb"

Model-based Systems Engineering



Ontology Based Systems Engineering


Fahrdienstleiter1
ZLiC
ZF
Fahrdienstleiter2
Wird Zug 4711 angenommen?
Zug 4711 (bis Jänschwalde) ja.
Ankunftsmeldung
Zug 4711 in Jänschwalde.
Ich wiederhole. Zug 4711 in Jänschwalde.
Ja. Richtig.
Fahrerlaubnis
Darf Zug 4711 bis Schwarze Pumpe fahren?
alt
[Fahrstrasse eingestellt]
Zug 4711 darf bis Schwarze Pumpe fahren.
Ich wiederhole. Zug 4711 darf bis Schwarze Pumpe fahren.
Nein. Warten.
Ankunftsmeldung
Zug 4711 in Schwarze Pumpe.
Ich wiederhole. Zug 4711 in Schwarze Pumpe.
Ja. Richtig.
Wird Zug 4712 angenommen?
Zug 4712 ja.
20 Minuten später
Zug 4712 in Jänschwalde.
Application
Middleware
Virtualization
Node 1
Node 2
. . .
Node n
Infrastructure
VerifiedPosition
Init
inMovement
e==1
e=0
e==2
e=0
Components
«subsystem» Festdatenbalise
«subsystem» Zug
«person» Triebfahrzeugfuehrer
«component» TF-Tablet
«subsystem» Zugleiter in the Cloud
«component» Notfunk-System
«component» Digitales Belegblatt
Digitale Leit- und Sicherungstechnik
«component» Stellwerkslogik
«subsystem» Weiche
«component» ObjectController
«interface» NFC
«interface» 868Mhz/GSM-R
«interface» gRPC/TLS
«interface» EULYNXoverRaSTA

EULYNX-based Direct Traffic Control

Communication Protocol

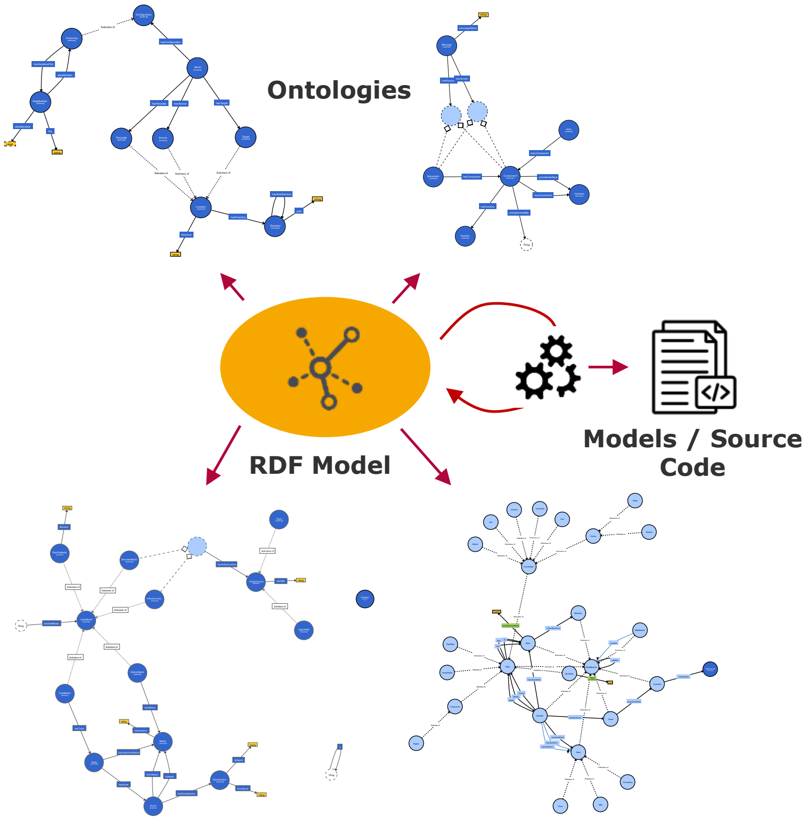
Lightweight RDF Model

Code Generation

Simulation

Documentation

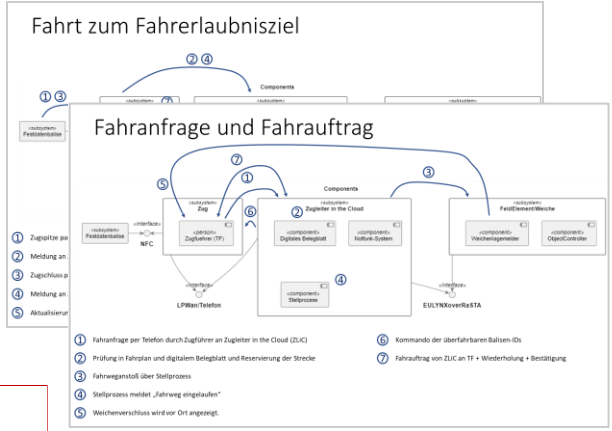
Micromodels and Transformations



- Clear and precise ontologies for each aspect
  - Components
  - Data model
  - Configuration
- Reuse existing ontologies
  - State machines
- Homogeneous and linkable data models for contextual relations
- Transformations to other models and formats or source code

Component Model

- Validation of model by domain experts
- Middleware for message exchange (DDS)
- Virtualization through Kubernetes or docker-compose
- Pipeline for build, test and deployment



① Fahranfrage per Telefon durch Zugführer an Zugleiter in the Cloud (ZLiC)

② Prüfung in Fahrplan und digitalem Belegblatt und Reservierung der Strecke

③ Fahrweganstoß über Stellprozess

④ Stellprozess meldet „Fahrweg eingelaufen"

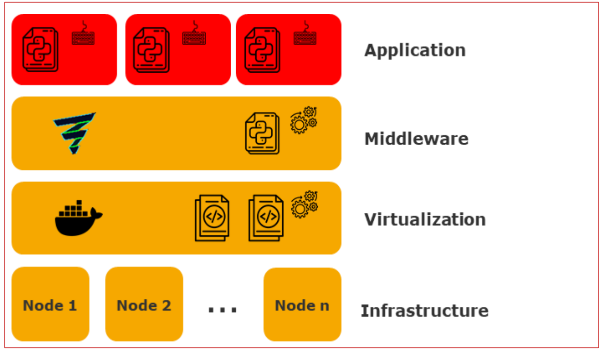
⑤ Weichenverschluss wird vor Ort angezeigt.

⑥ Kommando der überfahrbaren Balisen-IDs

⑦ Fahrauftrag von ZLiC an TF + Wiederholung + Bestätigung

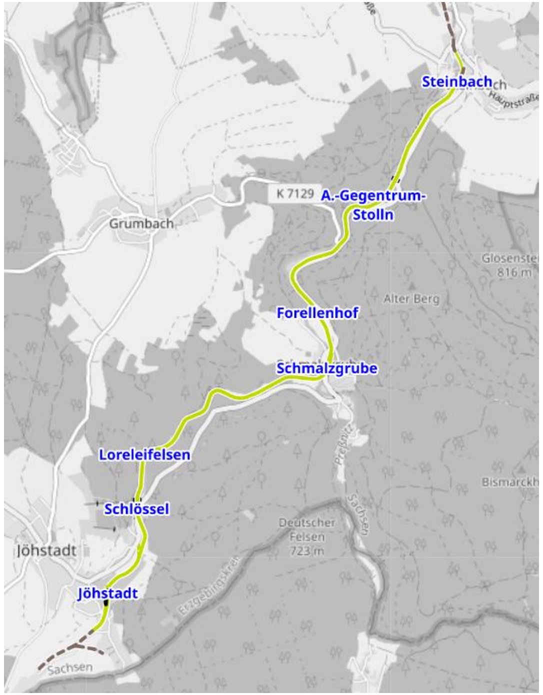
Visual Representation for Validation

Pipeline

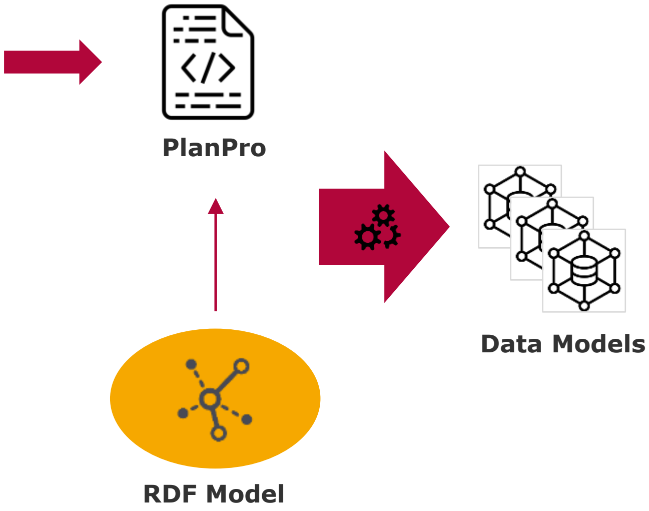


Software Stack

Data Models



OpenStreetMap



PlanPro

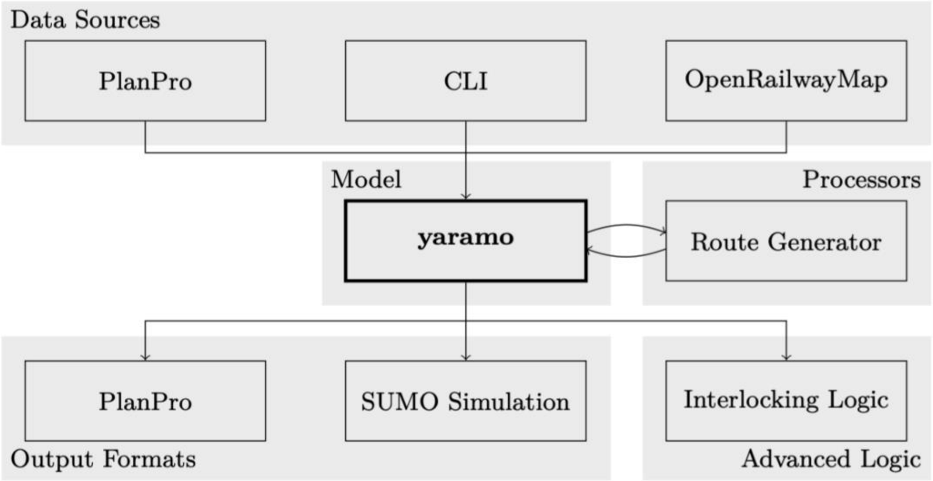
RDF Model

Data Models

- Interlocking logic
  - PlanPro model
- Digital „Belegblatt“
  - Stations
  - Routes
- NLU
  - Station names
- Simulation
  - Signals and Points

yaramo

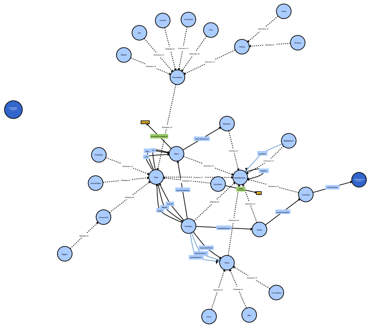
- The yaramo-model is a lightweight model for railway infrastructure
- Focus on simulation and test automation



- Model extraction from several data sources
- Processors to modify the model
- Exports and Logics as consumers of model instances

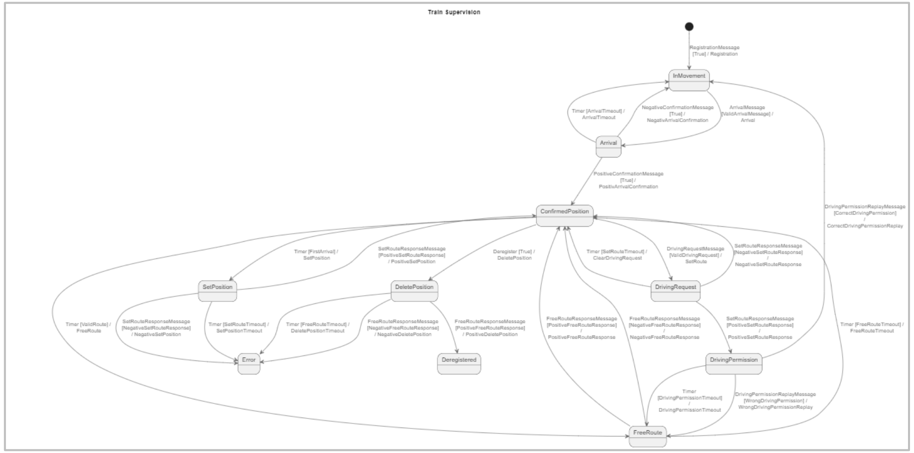
- Available on GitHub: https://github.com/simulate-digital-rail/yaramo

Communication State Machine



State Machine Ontology

RDF Model


Train Supervision
RegistrationMessage [True] / Registration
InMovement
Timer [ArrivalTimeout] / ArrivalTimeout
NegativeConfirmationMessage [True] / NegativArrivalConfirmation
ArrivalMessage [ValidArrivalMessage] / Arrival
Arrival
PositiveConfirmationMessage [True] / PositivArrivalConfirmation
ConfirmedPosition
DrivingPermissionReplayMessage [CorrectDrivingPermission] / CorrectDrivingPermissionReplay
Timer [FirstArrival] / SetPosition
SetRouteResponseMessage [PositiveSetRouteResponse] / PositiveSetPosition
Deregister [True] / DeletePosition
Timer [SetRouteTimeout] / ClearDrivingRequest
DrivingRequestMessage [ValidDrivingRequest] / SetRoute
SetRouteResponseMessage [NegativeSetRouteResponse] / NegativeSetRouteResponse
SetPosition
DeletePosition
DrivingRequest
Timer [ValidRoute] / FreeRoute
SetRouteResponseMessage [NegativeSetRouteResponse] / NegativeSetPosition
Timer [SetRouteTimeout] / SetPositionTimeout
Timer [FreeRouteTimeout] / DeletePositionTimeout
FreeRouteResponseMessage [NegativeFreeRouteResponse] / NegativeDeletePosition
FreeRouteResponseMessage [PositiveFreeRouteResponse] / PositiveDeletePosition
FreeRouteResponseMessage [PositiveFreeRouteResponse] / PositiveFreeRouteResponse
FreeRouteResponseMessage [NegativeFreeRouteResponse] / NegativeFreeRouteResponse
SetRouteResponseMessage [PositiveSetRouteResponse] / PositiveSetRouteResponse
Timer [FreeRouteTimeout] / FreeRouteTimeout
Error
Deregistered
DrivingPermission
Timer [DrivingPermissionTimeout] / DrivingPermissionTimeout
DrivingPermissionReplayMessage [WrongDrivingPermission] / WrongDrivingPermissionReplay
FreeRoute

Documentation

Final state machine

Generated Code

Digital „Belegblatt“

Interlocking Logic

Interlocking Logic

- Generic interlocking logic based on a yaramo topology
- Supports:

Infrastructure Status

Generation of Overlaps

Connections to real hardware and simulations

Determination of Infrastructure Changes

Asynchronous and timed behaviour

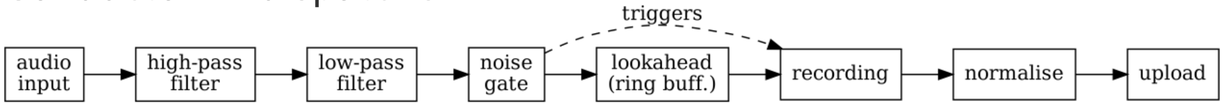
Evaluation of infrastructure changes

Flank Protection (in progress)

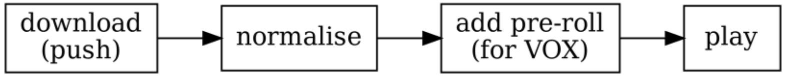
- Available on GitHub: https://github.com/simulate-digital-rail/interlocking

RadioIO

- Between digital dispatcher ("Zugleiter") and conductor ("Zugführer")
  - Communication via radio, other technology conceptually possible
  - Conductor → dispatcher



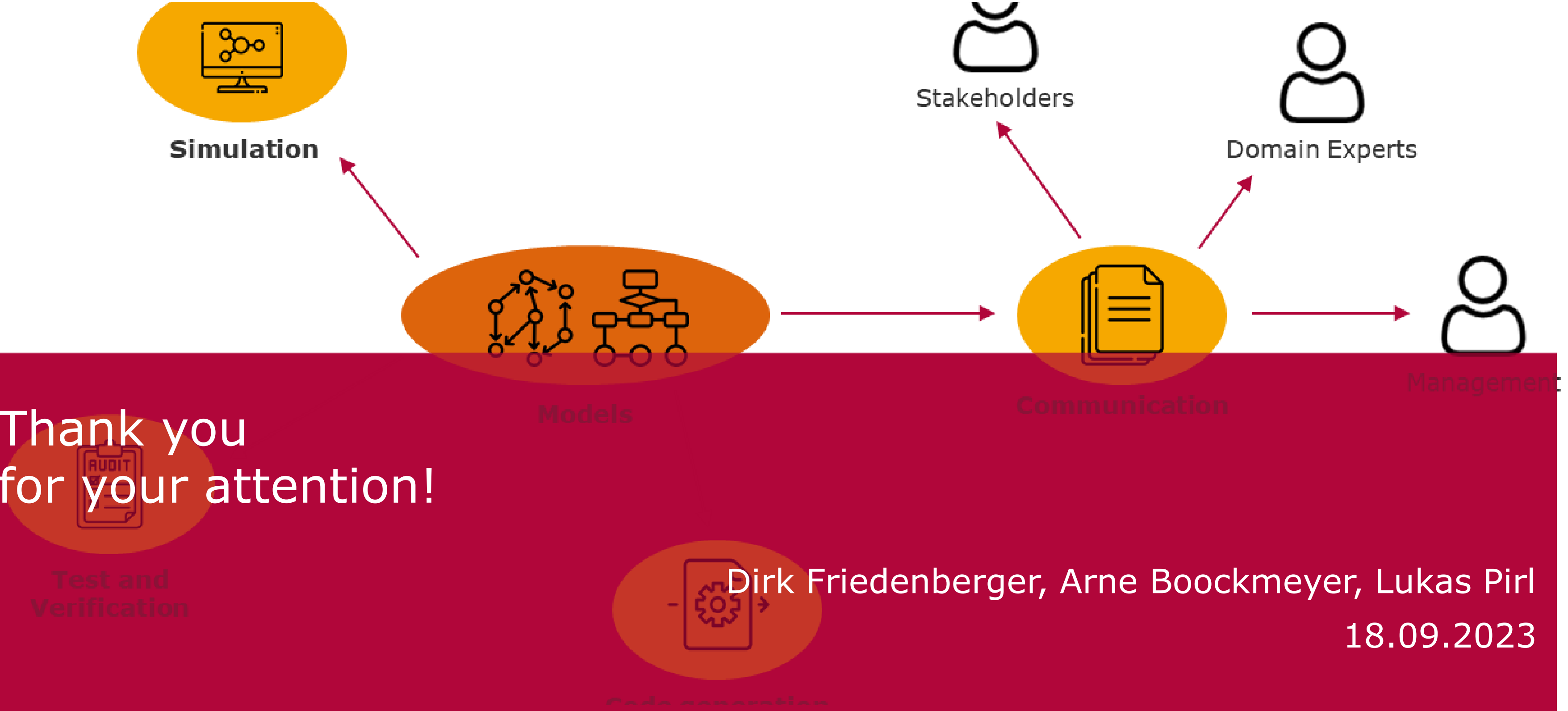
  - Dispatcher → conductor



- Custom implementation
  - Mostly pre-allocated & zero-copy, DIY DSP (no framework found)
  - Nim (programming language) → C → gcc → machine code

Future Work

- Safety considerations
- Emergency stop order
- Model Validation



Thank you
for your attention!

Dirk Friedenberger, Arne Boockmeyer, Lukas Pirl

18.09.2023