

13 June 2025

Online Integration of mobiTopp and MATSim for Agent-Based Simulations of On-Demand Services (and beyond)

Robin Andre, KIT

Nico Kuehnel, MOIA

Gabriel Wilkes, KIT

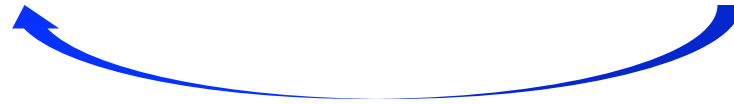
Martin Kagerbauer, KIT

Peter Vortisch, KIT

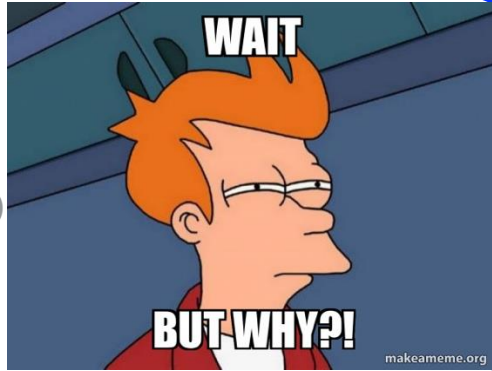
MOIA

mobiiopp

MATSim
Multi-Agent Transport Simulation



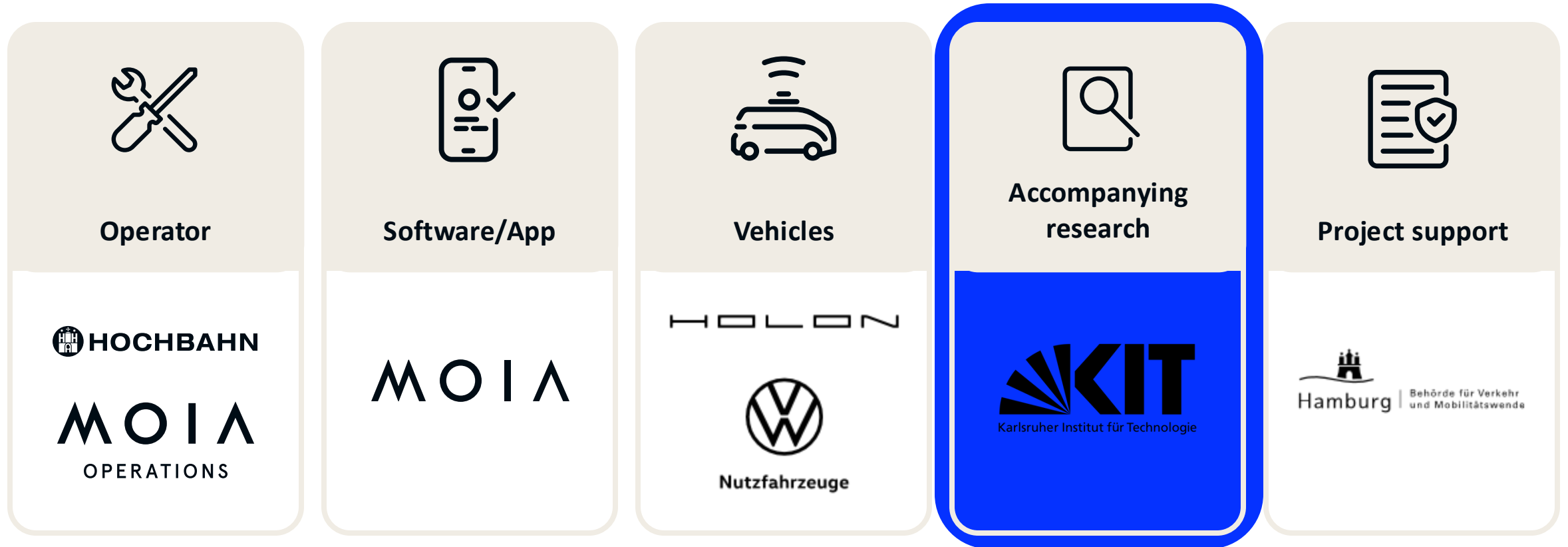
mobiiopp



MATSim
Multi-Agent Transport Simulation

The ALIKE project unites seven partners in their expertise in **bringing autonomous ridepooling to Hamburg's streets**

MOIA





Autonomous driving functions



Digital booking



Technical supervision



Acceptance



Structure of the overall system



Accessibility

ALIKE introduces 2 AD-Vehicles in a testbed to Hamburg streets

VW ID.Buzz AD



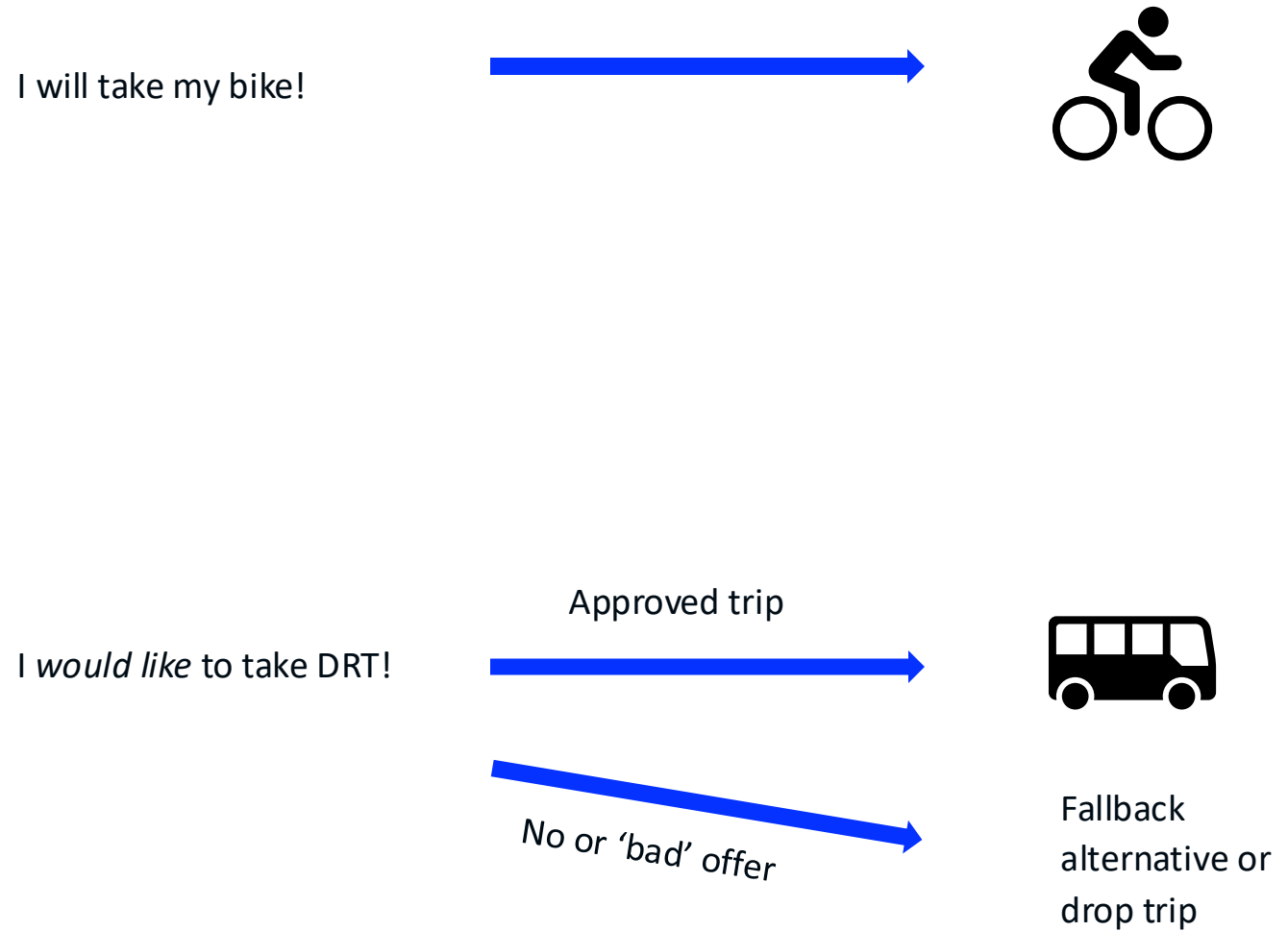
- Up to 10 prototype vehicles
- Up to 4 passengers, SDS von Mobileye

HOLON Mover



- Up to 10 prototype vehicles
- Up to 15 passengers, SDS von Mobileye

On-demand transport may be a bit more complex than other modes



On-demand transport may be a bit more complex than other modes

I will take my bike!



Decision usually taken between iterations in MATSim!

I would like to take DRT!

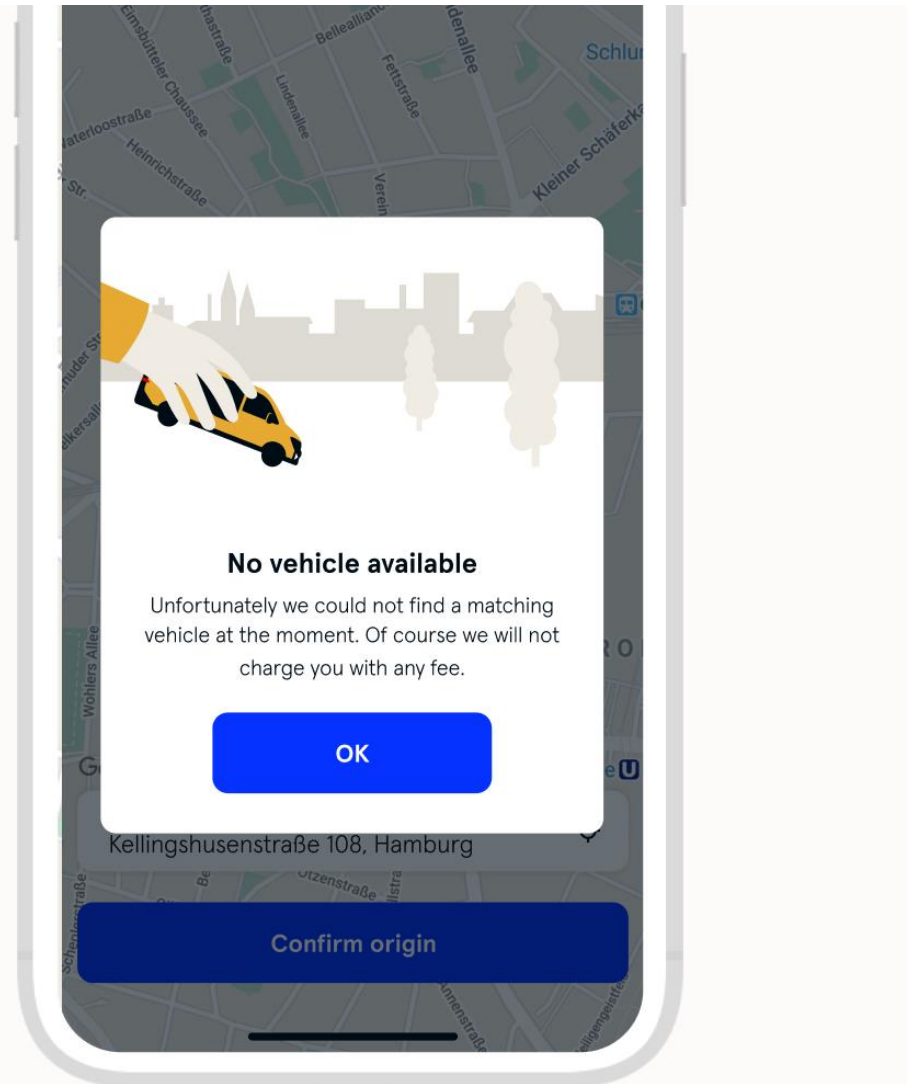
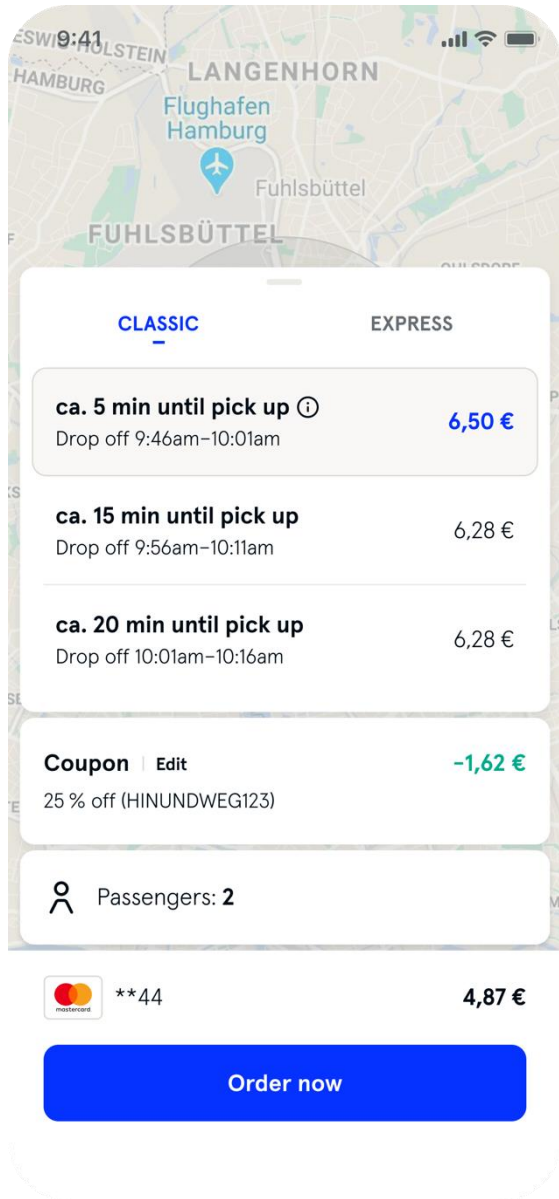
Approved trip



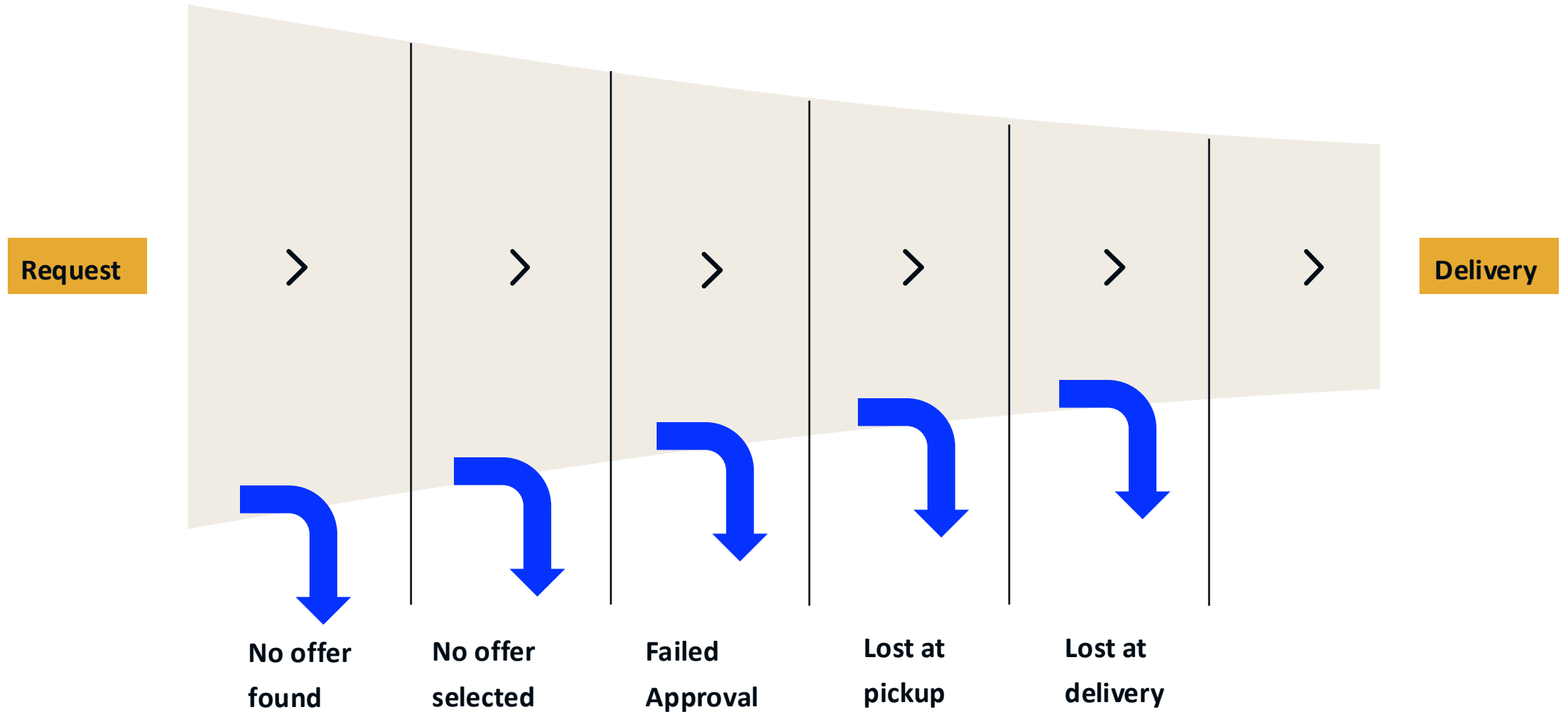
No or 'bad' offer



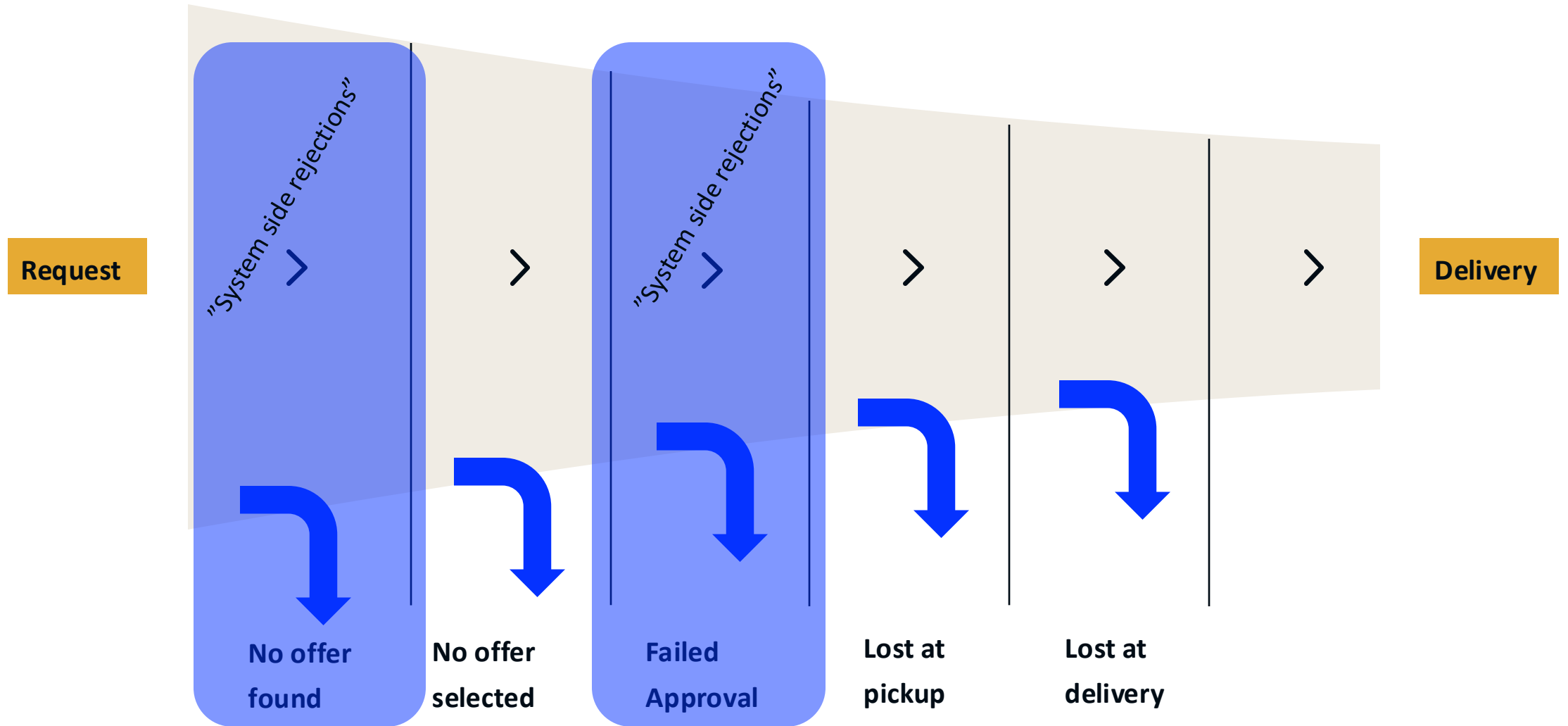
Fallback alternative or drop trip



A ride request may fail at multiple steps

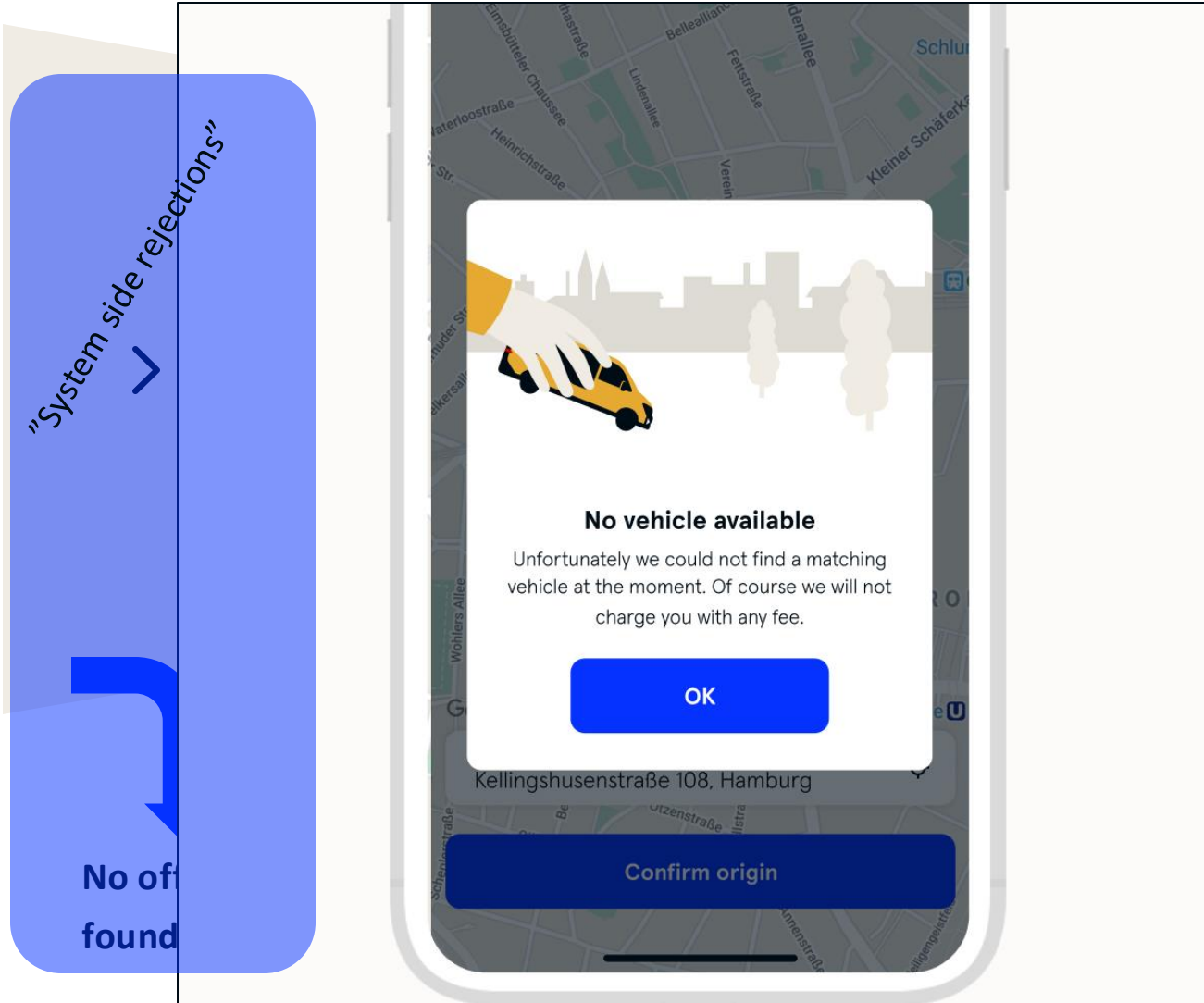


Default MATSim DRT only handles system side rejections



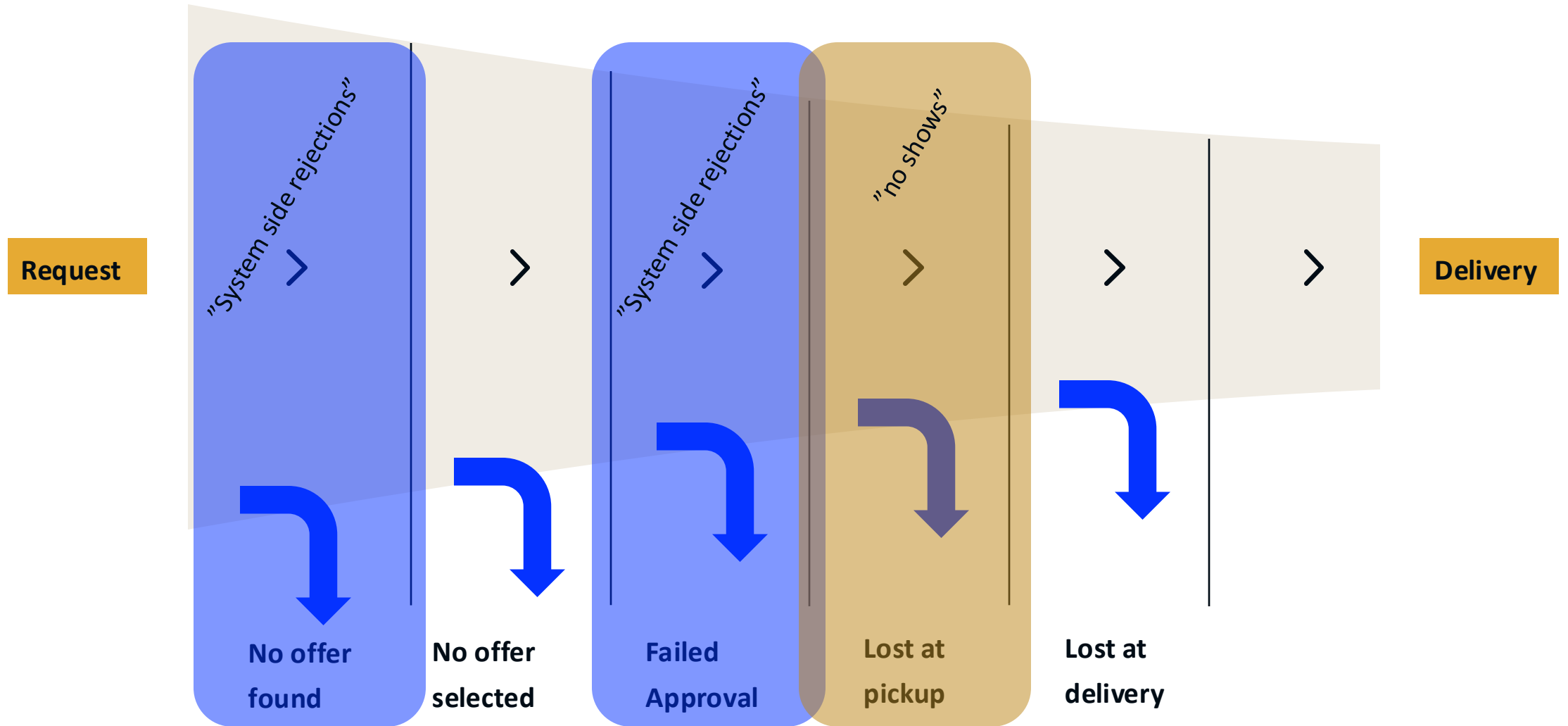
Default MATSim DRT only handles system side rejections

Request

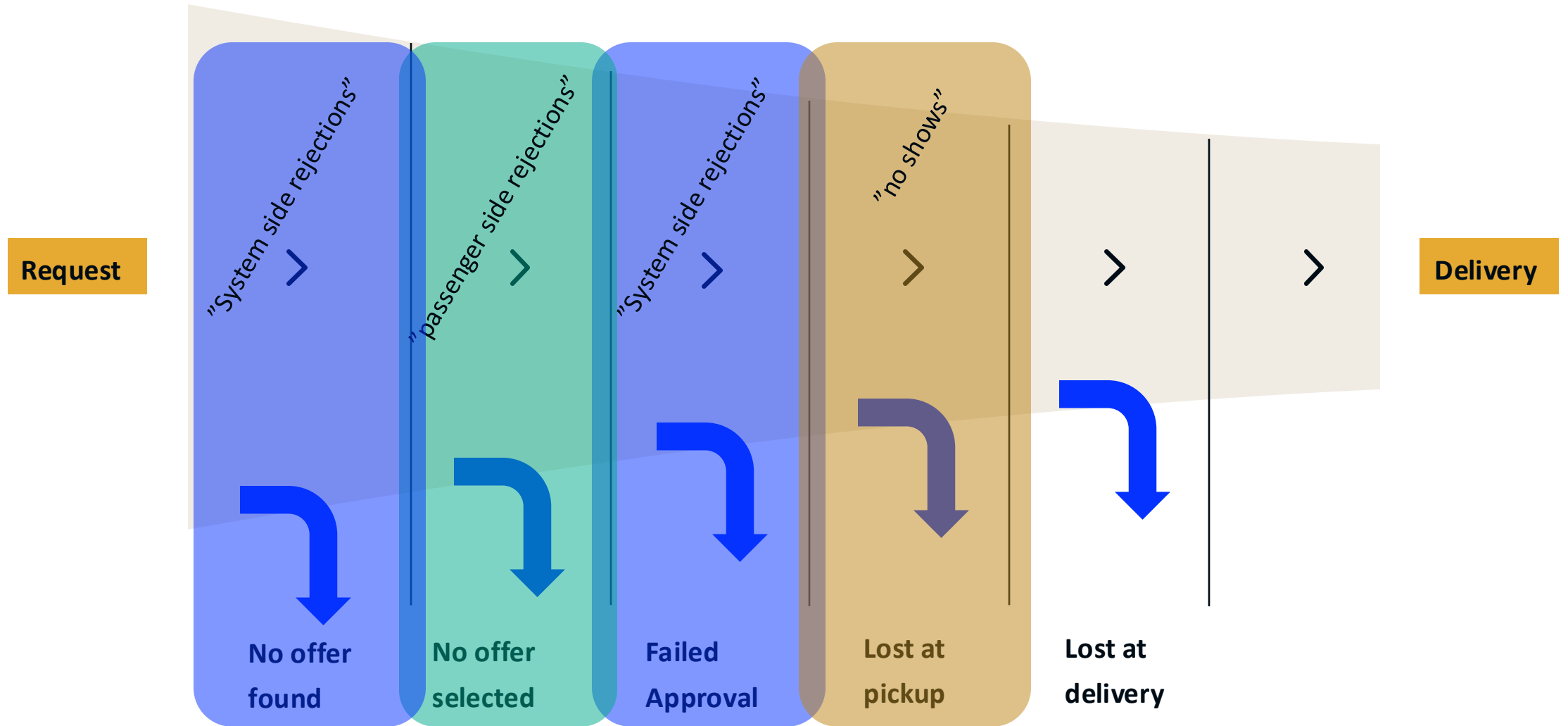


Delivery

The prebooking extension can handle „no shows“

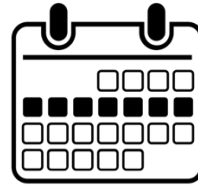


There is a need to model passenger side rejections

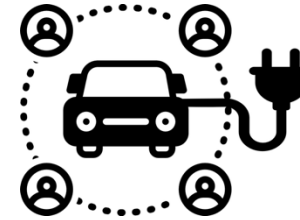


mobiTopp

- Agent- and activity-based travel demand modeling framework
- Open source, written in Java, currently re-engineered in Kotlin
- Split into:
 - “Long term” module (population synthesis etc.)
 - “short term” module (chronological simulation)
- Models with more than 3 million agents
- Feasible simulation runtimes with 100% population



Consistent behavior of people and households throughout one week



Individual private and shared vehicles with inter-dependencies



Ownership models for car, transit pass, mobility services, ...



Integration of numerous influences on mobility behavior (discrete choice models)

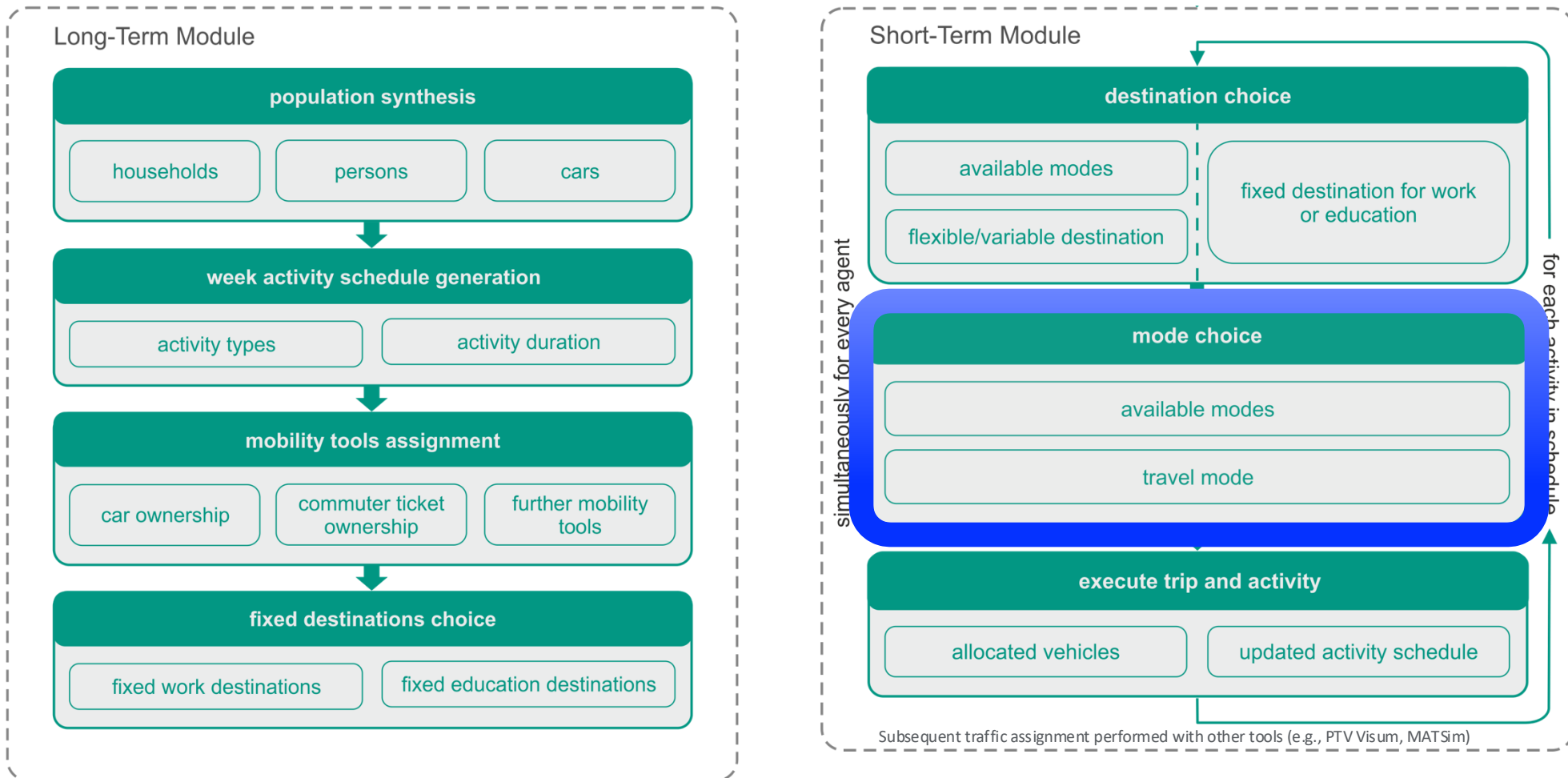


Changes in the choice activities' locations due to e.g., increase of working from home



Combining several modes on one route (intermodal trips)

mobiTopp



Illustrative communication

- Currently `_every_` agent sends a request
- For the future, we aim for finding “interested candidates” that actually consider DRT
- Agents need to be able to choose a fallback mode (from a reduced choice set?)
- From real world MOIA data we can already estimate selection models based on shown offers

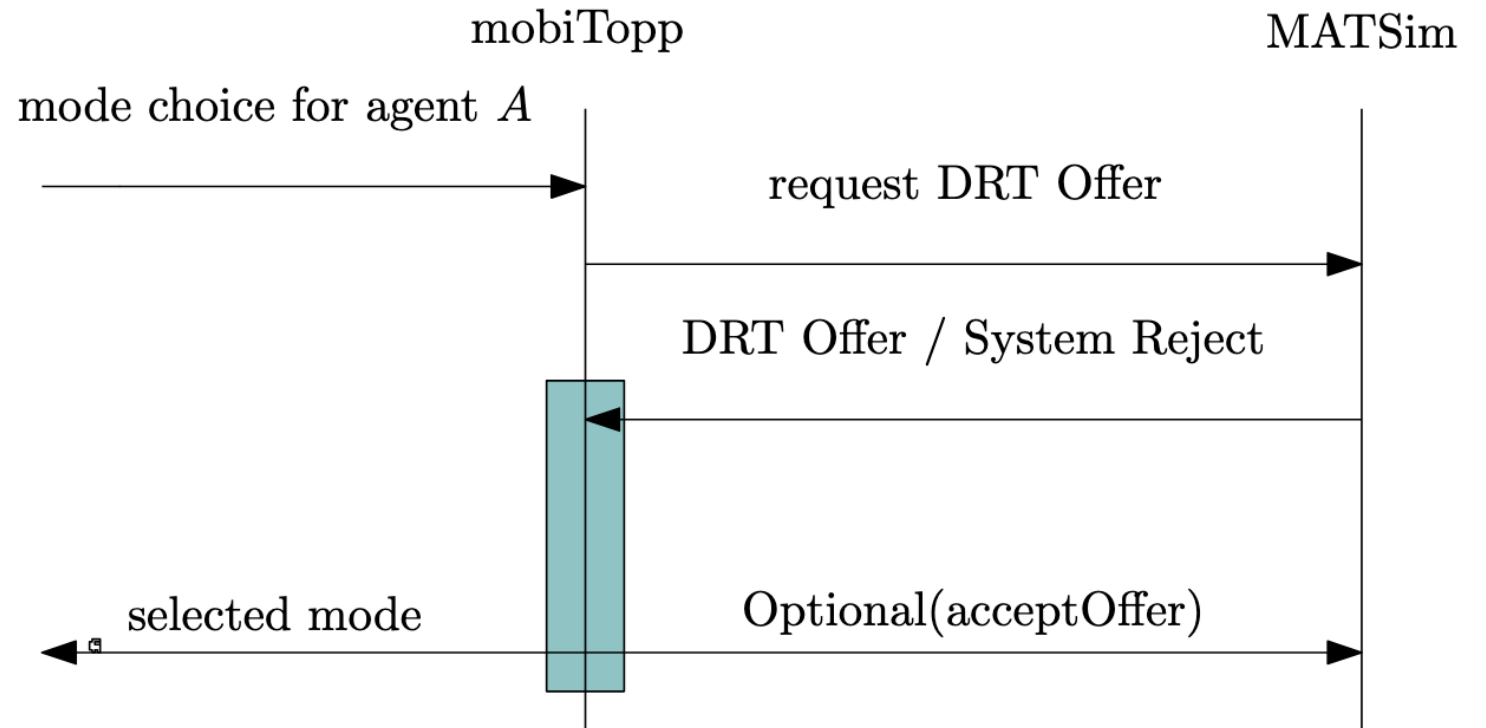


Figure 1: Example communication between mobiTopp and MATSim

Trip based coupling

- Technically, for each trip,
 1. the agent is cloned as a MATSim agent
 2. A plan is created with a single leg connecting origin and destination
 3. MATSim's trip router is executed on the clone
 4. The agent is inserted into the mobsim immediately
- mobiTopp and MATSim are currently coupled at minute resolution

Test Scenario

- Small test scenario in Hamburg
- ~13k agents
- ~278k trips over one week
- bicycle, pt, car, car pass., walk, drt
- Two vehicles, five stops
- 87k rides without live coupling
- 16k rides with live coupling

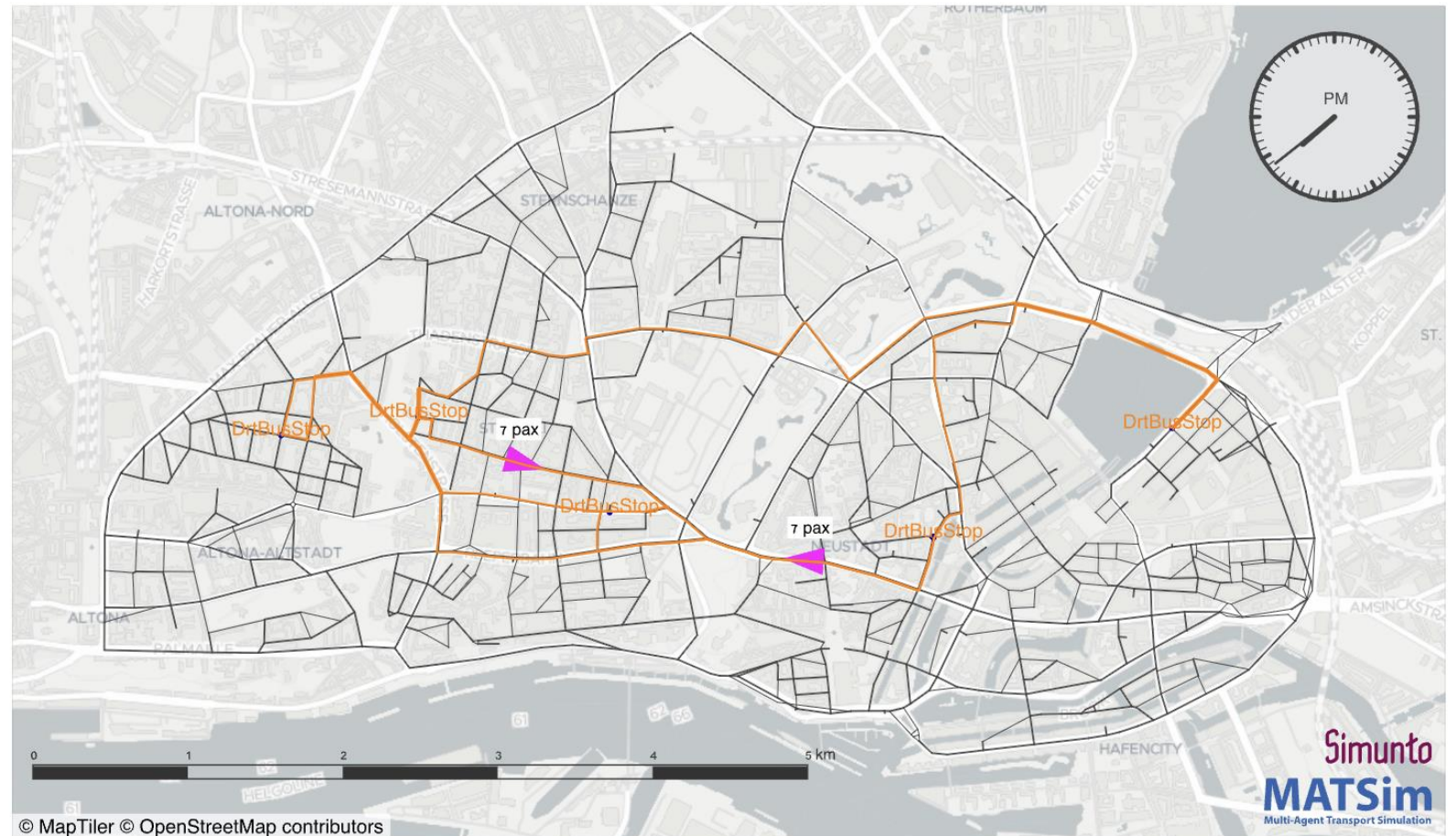


Figure 2: Visualization of the example scenario with five stop locations between which agents request rides.

Test Scenario

The setup allows us to distinguish between user and system rejections

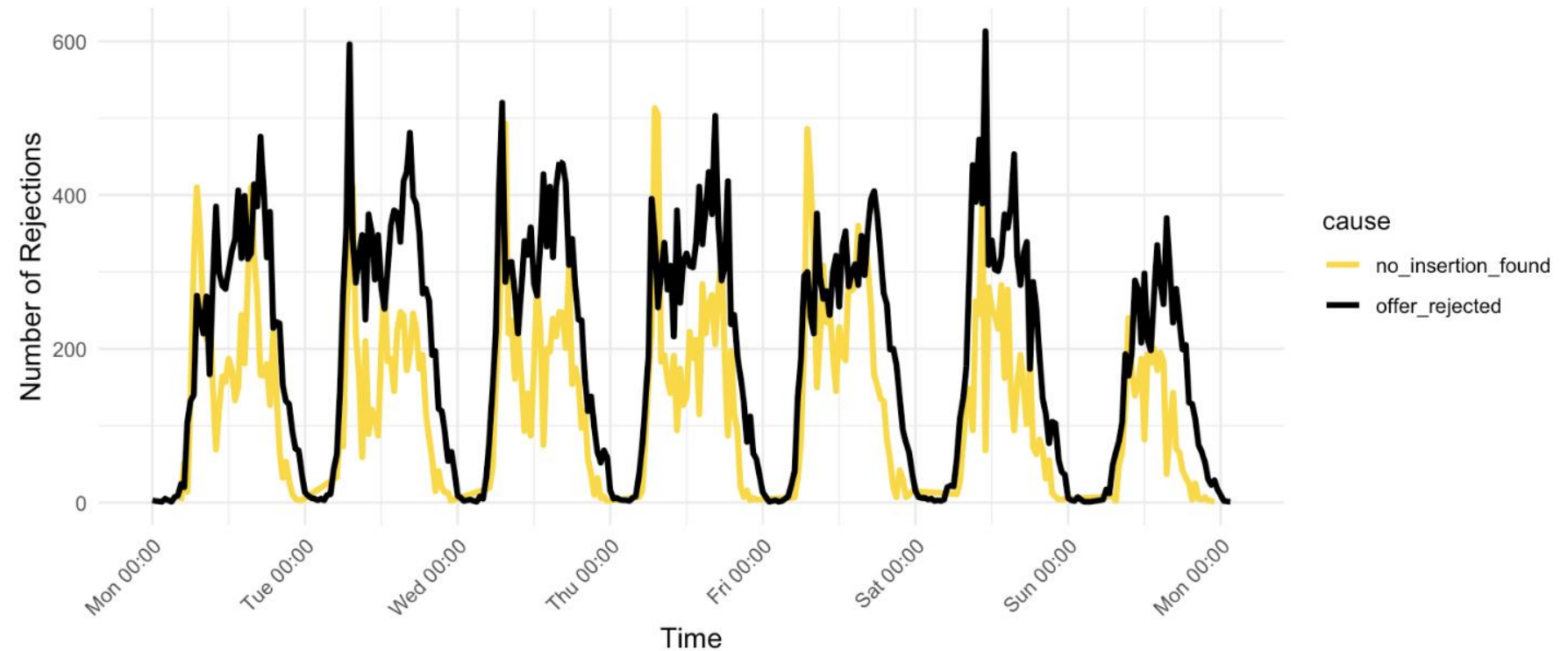


Figure 3: System-side and customer-side rejections over the course of the week.

Outlook

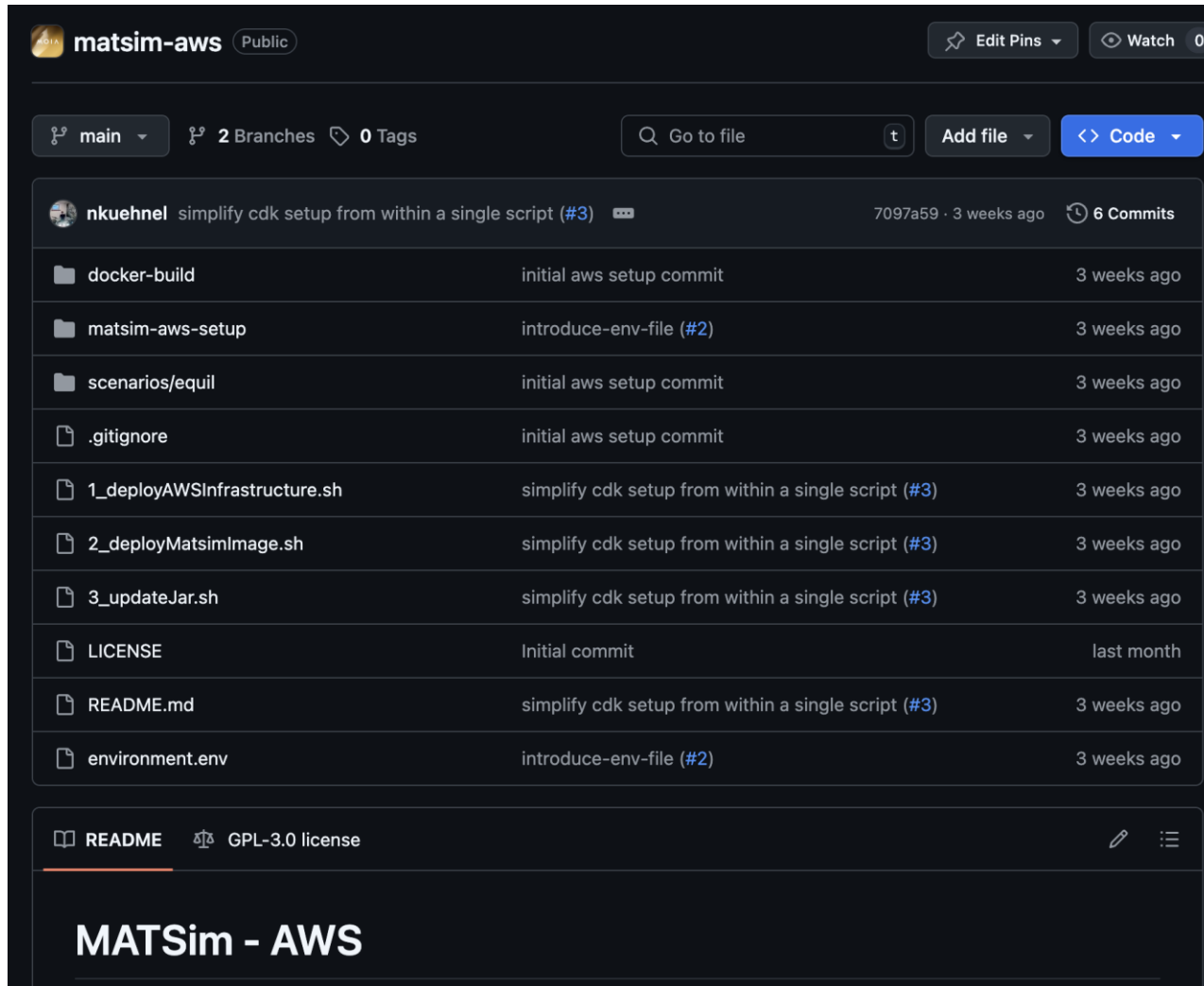
- Full behavioral model yet to be defined
- The same approach could be extended to virtually any mode
 - Live PT-Routing based on occupancy and delays
 - Live car-routing based on current congestion levels (may need pre-iterated traffic levels)
 - Etc...
- Performance is a bottleneck, the synchronization between mobiTopp and MATSim is currently under investigation
- Change from trip-based MATSim clones to fully sync'ed agents (propagate delays and and ensure time-space consistency)

Thank you

Nico Kuehnel
nico.kuehnel@moia.io

MOIA

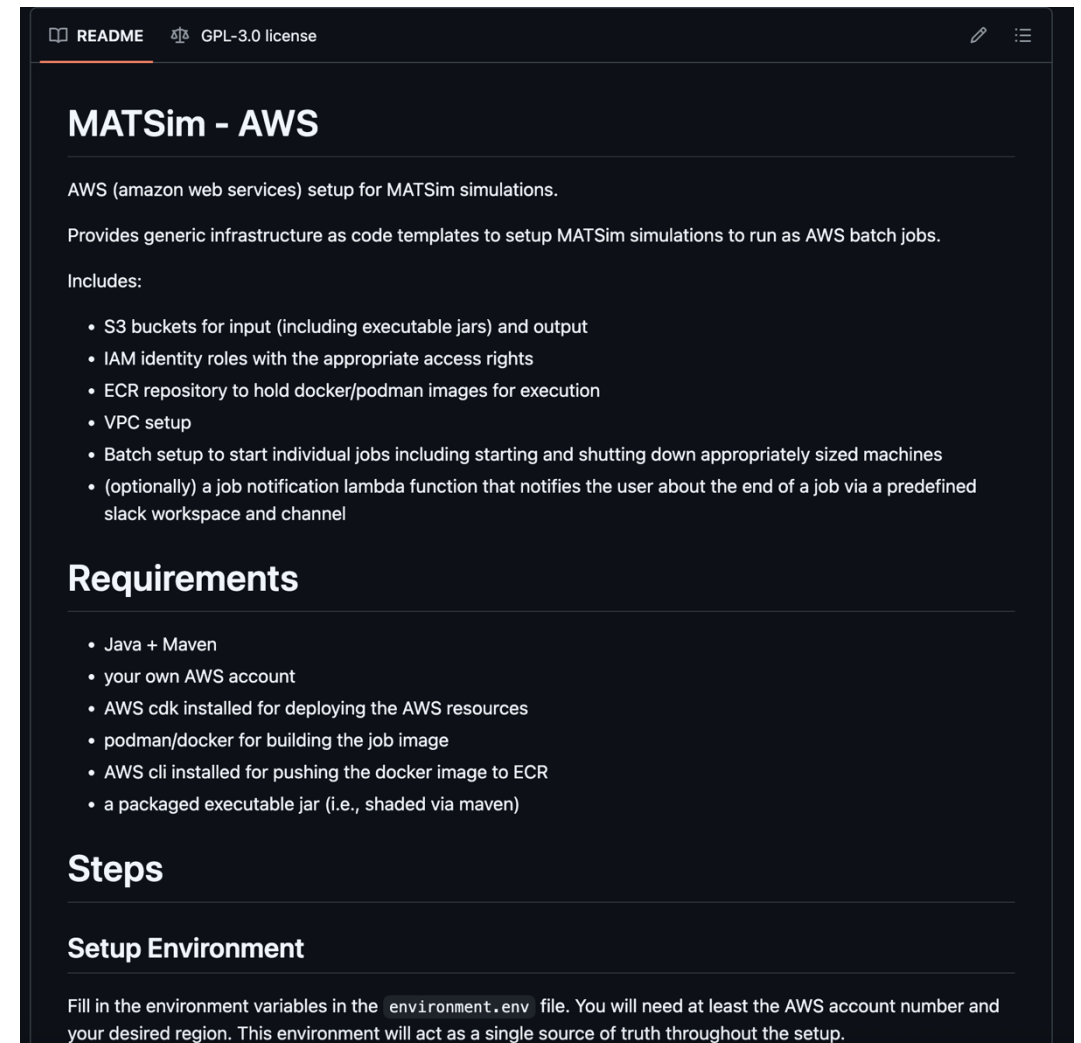
<https://github.com/moia-oss/matsim-aws>



The screenshot shows the GitHub repository page for `matsim-aws`. The repository is public and has 2 branches and 0 tags. The main branch is selected. The repository description is "simplify cdk setup from within a single script (#3)". The repository has 7097a59 commit and 6 commits. The repository includes the following files and folders:

File/Folder	Commit Message	Commit Date
docker-build	initial aws setup commit	3 weeks ago
matsim-aws-setup	introduce-env-file (#2)	3 weeks ago
scenarios/equil	initial aws setup commit	3 weeks ago
.gitignore	initial aws setup commit	3 weeks ago
1_deployAWSInfrastructure.sh	simplify cdk setup from within a single script (#3)	3 weeks ago
2_deployMatsimImage.sh	simplify cdk setup from within a single script (#3)	3 weeks ago
3_updateJar.sh	simplify cdk setup from within a single script (#3)	3 weeks ago
LICENSE	Initial commit	last month
README.md	simplify cdk setup from within a single script (#3)	3 weeks ago
environment.env	introduce-env-file (#2)	3 weeks ago

The repository also includes a README file and a GPL-3.0 license.



The screenshot shows the README file for the `matsim-aws` repository. The README is titled "MATSim - AWS" and is licensed under GPL-3.0. The README describes the project as "AWS (amazon web services) setup for MATSim simulations." and provides generic infrastructure as code templates to setup MATSim simulations to run as AWS batch jobs. The README includes the following sections:

Includes:

- S3 buckets for input (including executable jars) and output
- IAM identity roles with the appropriate access rights
- ECR repository to hold docker/podman images for execution
- VPC setup
- Batch setup to start individual jobs including starting and shutting down appropriately sized machines
- (optionally) a job notification lambda function that notifies the user about the end of a job via a predefined slack workspace and channel

Requirements

- Java + Maven
- your own AWS account
- AWS cdk installed for deploying the AWS resources
- podman/docker for building the job image
- AWS cli installed for pushing the docker image to ECR
- a packaged executable jar (i.e., shaded via maven)

Steps

Setup Environment

Fill in the environment variables in the `environment.env` file. You will need at least the AWS account number and your desired region. This environment will act as a single source of truth throughout the setup.