# Agent-based Simulation Testbed for On-demand Transport Services

# (Demonstration)

Michal Čertický, Michal Jakob, Radek Píbil, Zbyněk Moler
Agent Technology Center, Faculty of Electrical Engineering,
Czech Technical University, Prague, Czech Republic
{certicky, jakob, pibil, moler}@agents.fel.cvut.cz

## ABSTRACT

We present an open-source simulation testbed for the development, comparison and analysis of on-demand transport services. The testbed is designed to evaluate the performance of agent-based, on-demand transport vehicle allocation and routing mechanisms; accounting for different vehicle fleets, road network topologies and transport demand structures. A wide range of metrics, including passenger waiting times, vehicle occupancy, or distance travelled, can be used for measuring system performance. In order to encourage the comparison of different allocation and routing mechanisms under standard conditions, the testbed comes with a predefined set of ready-to-use benchmarking scenarios.

## Categories and Subject Descriptors

I.6 [**Simulation and Modelling**]: Applications

## General Terms

Experimentation, Algorithms

## Keywords

agent-based simulation, modelling, transport, testbed

## 1. INTRODUCTION

*On-demand transport systems* promise significant improvements in personal mobility due to more efficient and responsive utilization of available transport vehicles. In on-demand transport systems, vehicle routes and schedules are not fixed a priori. Instead, they are dynamically adapted to best serve continuously incoming transport requests. While in the past on-demand transport was used mainly for providing small-scale specialized paratransit services, it is now increasingly utilised as the basis of various general-purpose real-time ridesharing, taxi or bus-on-demand services. Since on-demand transport systems are inherently multi-agent, agent-based allocation and coordination techniques have been increasingly and successfully employed to solve them [1].

**Appears in:** *Alessio Lomuscio, Paul Scerri, Ana Bazzan, and Michael Huhns (eds.), Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), May 5-9, 2014, Paris, France.*

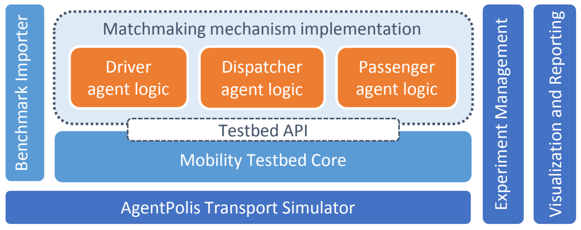

**Figure 1:** Overview of testbed architecture.

The performance of an on-demand transport service depends crucially on two factors: (1) the *matchmaking mechanism* used to allocate the vehicles to passengers and to determine vehicle routes; and (2) parameters of the *deployment scenario*, in particular the topology of the underlying road network and spatio-temporal structure of transport demand. Understanding how these factors affect the on-demand transport performance is essential for principled development and deployment of on-demand transport services. Due to the complex nature of on-demand transport service systems, gaining such understanding is difficult without appropriate simulation modelling and benchmarking tools.

So far the availability of such tools has been very limited. Within the broad family of *pickup and delivery problems*, benchmarking suites only exist for static versions of freight transport vehicle routing problems[1]. To the best of our knowledge, no benchmarking tools exist for dynamic, passenger-oriented variants of pickup and delivery problems.

To fill this gap, we have developed an open-source *simulation testbed*[2] that leverages the fully agent-based transport modelling approach [2] implemented by the AgentPolis transport-oriented simulation framework [3].

## 2. TESTBED ARCHITECTURE

The components of the testbed can be broadly divided into three layers (see Figure 1):

**AgentPolis Transport Simulator:** The simulator provides core simulation engine, based on the discrete-event simulation approach, and a basic transport domain model. The transport domain model implements the model of the individual elements of the transport system, such as road networks and vehicles, and the behaviour logic associated with them. It also provides routing algorithms and communication interfaces designed to simplify the implementation of higher-level agent control logic.

[1] `http://www.or.deis.unibo.it/research_pages/ORinstances/VRPLIB/VRPLIB.html`
[2] `http://github.com/agents4its/mobilitytestbed`

**Testbed Core:** The core specializes general AgentPolis simulator for the specific purpose of modelling on-demand transport services. It implements the model of the Passenger, Vehicle and Dispatcher agents and provides extensible abstractions for defining their behaviour (see Section 3).

**Matchmaking Mechanism:** User-supplied implementation of a specific matchmaking mechanism the user wants to experimentally evaluate.

In addition to the above, the testbed provides a suite of tools that facilitate creation, execution and evaluation of simulation experiments.

## 3. MATCHMAKING MECHANISM

The most complicated part of the evaluation of a matchmaking mechanism is its implementation and integration with the simulation. Our testbed greatly simplifies this task by providing a clearly defined integration interface. Specifically, the user only needs to implement a few selected methods of the `DriverLogic`, `PassengerLogic` or `DispatchingLogic` abstract classes that govern the behaviour of the Driver, Passenger and Dispatcher agents, respectively.

The `DriverLogic`'s `ProcessNewRequest` method is invoked when a Driver receives a transport request from a Passenger. The `PassengerLogic`'s `ProcessNewProposal` method is called when a Passenger receives a trip proposal from a Driver or Dispatcher specifying proposed trip arrangements (in particular time and price). Finally, the `processNewAcceptance` and `processNewRejection` methods can be implemented by a Driver or Dispatcher to handle passenger's response to earlier trip proposals.

While implementing a matchmaking logic, the user utilizes the *Testbed API* to access the state of the simulation (for example by calling the `getCurrentPosition` function to locate certain vehicles) and to perform the intended behaviour by executing specific actions (such as driving to a certain preplanned location by calling the `driveNextPartOfTripPlan` function of the `DriverLogic` class).

Altogether, the testbed provides sufficiently flexible abstractions to implement and evaluate a wide variety of matchmaking mechanisms, from fully centralized, through hybrid, to fully distributed mechanisms.

## 4. EXPERIMENT WORKFLOW

The typical workflow for the experimental evaluation of a matchmaking mechanism consists of three steps:

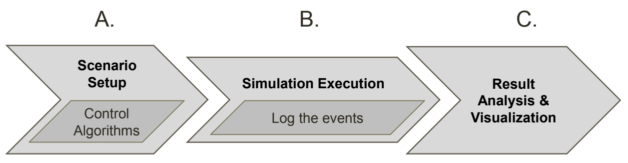

**Figure 2:** Three-step experiment process.

**Step A – Experiment Specification:** Besides the implementation of a matchmaking mechanism, the user needs to provide a *benchmark scenario package* defining a specific instance of an on-demand transport system on which the mechanism is to be evaluated. The scenario package consists of an OpenStreetMap (OSM) *map file* covering the area of interest and two JSON files containing the specification of the *Driver agents*, along with their vehicle parameters and initial locations, and the transport demand expressed as an enumeration of all the *Passenger agents* with their mobility needs and special requirements.

**Step B – Simulation Execution:** Once the matchmaking mechanism is implemented and the experiment scenario specified, the user runs a simulation or a batch of simulations using the testbed's experiment management tools. The simulator generates detailed event logs that capture the progress of each simulation run. The user can observe and control simulation execution through built-in map-based and event-based graphical visualization interfaces.

**Step C – Result Analysis and Visualization:** After the simulation has finished, the testbed's reporting and visualization tools process low-level event logs and calculate higher-level, aggregated performance metrics. The built-in metrics include distance driven, fuel consumption, $CO_2$ emissions, passenger waiting time statistics, or the runtime of matchmaking algorithms. The user can also automatically run a number of scenario instances with varying parameters, in order to discover and visualize the influence of various situation properties on performance metrics (e.g. vehicle fleet size vs. total distance driven). Finally, the testbed converts the recorded event logs into a collection of KML files that can be visualized in an interactive geobrowser *Google Earth* to inspect the simulation runs in a fine spatial and temporal detail. The geospatial visualizations can also be used to show how the performance metrics vary across different parts of the experiment area (see Figure 3).

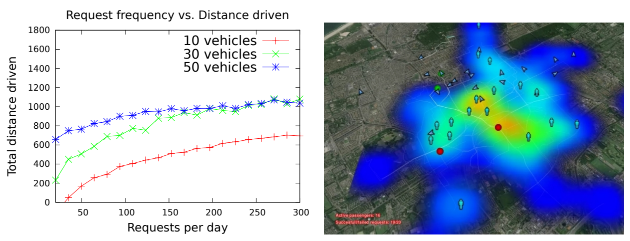

**Figure 3:** Outputs: Dependency of a perfomance metric on scenario paramteres (left) and on geospatial location (right).

## 5. ACKNOWLEDGEMENTS

This work was funded by the Ministry of Education, Youth and Sports of Czech Republic (grants no. TE01020155 and 7E12065) and by the European Union Seventh Framework Programme FP7/2007-2013 (grant agreement no. 289067).

## 6. REFERENCES

[1] A. Glaschenko, A. Ivaschenko, G. Rzevski, and P. Skobelev. Multi-agent real time scheduling system for taxi companies. In *Proceedings of AAMAS 2009*, 2009.

[2] M. Jakob and Z. Moler. Modular framework for simulation modelling of interaction-rich transport systems. In *Proceedings of IEEE ITSC 2013*, 2013.

[3] M. Jakob, Z. Moler, A. Komenda, Z. Yin, A. X. Jiang, M. P. Johnson, M. Pěchouček, and M. Tambe. AgentPolis: towards a platform for fully agent-based modeling of multi-modal transportation. In *Proceedings of AAMAS 2012-Volume 3*, 2012.