PART II

Extending MATSim

CHAPTER 5

Available Functionality and How to Use It

Andreas Horni and Kai Nagel

In this chapter you will learn about possibilities to extend and customize MATSim (Multi-Agent Transport Simulation) through provided functionality. In Chapter 45, you will see how you can hook your own extensions into MATSim.

5.1 MATSim Modularity

MATSim follows a modular concept, but a "module" is not a very specific term;[1] thus, modules can exist at many levels in a software framework. Also in MATSim, a range of different functionality types, such as config functions, replanning components, contributions, or even external tools,[2] are sometimes described as modules. Metaphorically speaking, a module can thus be seen as the greatest common divisor (gcd) of different functionality provided in MATSim. Much more important is understanding the different levels of access stemming from the generally modular architecture.

5.1.1 Levels of Access

MATSim currently provides five levels of access:

1. using the MATSim core only,
2. using the MATSim main distribution,

[1] According to the Merriam-Webster (`http://www.merriam-webster.com`), a module is "one of a set of parts that can be connected or combined to build or complete something" or more specifically "a part of a computer or computer program that does a particular job".
[2] Standalone tools referencing MATSim as a library, such as the network editor, or the visualizer Via.

How to cite this book chapter:
Horni, A and Nagel, K. 2016. Available Functionality and How to Use It. In: Horni, A, Nagel, K and Axhausen, K W. (eds.) *The Multi-Agent Transport Simulation MATSim*, Pp. 47–52. London: Ubiquity Press. DOI: http://dx.doi.org/10.5334/baw.5. License: CC-BY 4.0

3. using MATSim main distribution, contributions and possibly extensions,
4. writing "scripts in Java" and finally
5. writing your own extensions.

5.1.1.1 Using the Core Only

To use only the core, one needs to do the following (see Section 2.1):

- Download a MATSim release or a nightly build, by following the respective links at `http://matsim.org/downloads`.
- Obtain a network file and an initial plans file. Small versions can be typed by hand; larger versions should be generated automatically by some computational method.
- Write or edit a config file.
- Click on the MATSim jar file[3] and follow the instructions.

We think that the MATSim core is already quite powerful; for example, synthetic persons already follow full daily plans with a full daily scoring function; thus, opening times for activity types, departure time choice and schedule delay can be investigated.

5.1.1.2 Using MATSim Main Distribution

The extensions in the MATSim main distribution are, by design, very close to the MATSim core, thus requiring even less configuration than for contributions, as shown below. Often, providing additional files together with a respective config file entry is sufficient to use them; required steps are described below, case by case. Extensions contained in the main distribution are listed in a separate section at `http://matsim.org/extensions`.

5.1.1.3 Using One or More Contribs or Other Extensions

Contributions are in a separate part of the repository, separate from the MATSim main distribution. The documentation is not yet fully organized; information about contributions and other extensions can be found at `http://matsim.org/extensions`. For the contributions, there are also release versions and nightly builds, which can be found by following the links at `http://matsim.org/downloads`.

In general, contributions should provide main methods for use. We may eventually provide clickable jar files here as well, but for the time being, contributions need to be bundled with core MATSim (and potentially other contributions). As shown at `http://www.matsim.org/docs/extensions`, the syntax is roughly

```
java -Xmx2000m -cp MATSim.jar:contrib/contrib.jar org.matsim.contrib.run.RunXxx
   config.xml
```

where

- `-Xmx2000m` increases the Java heap space, so that most MATSim runs fit in,
- `MATSim.jar` needs to be replaced by a relative or absolute path to the MATSim jar to be used,
- `contrib/contrib.jar` needs to be replaced by a relative or absolute path to the contribution jar to be used,

[3] This has worked since winter 2014/15 and should be in the 0.8.x release.

- `org.matsim.contrib.run.RunXxx` needs to be replaced by the full Java class name containing the desired main method (given by the contribution documentation), and
- `config.xml` needs to be replaced by a relative or absolute path to a config file, which may contain additional sections specific to the contribution.

It is possible to combine several contributions in this way, provided someone has made a corresponding main method available. This can, in principle, be done relatively quickly, so those wishing to run studies with combinations of existing contributions, but without programming skills, can ask someone with those skills and with access to the repository for help.

5.1.1.4 Writing "Scripts in Java"

The contributions are written so that they can be plugged into MATSim via extension points (see Chapter 45). If a specific combination or configuration of modules is not (yet) available, one can write it. The syntax is roughly:

```
... main( ... ) {
    // construct the config object:
    Config config = ConfigUtils.xxx(...) ;
    config.xxx().setYyy(...) ;
    ...

    // load and adapt the scenario object:
    Scenario scenario = ScenarioUtils.loadScenario( config ) ;
    scenario.getXxx().doYyy(...) ; // (*)
    ...

    // load and adapt the controler object:
    Controler controler = new Controler( scenario ) ;
    controler.doZzz(...) ; // (**)
    ...

    // run the iterations:
    controler.run() ;
}
```

Extension points, especially at `(*)` and `(**)`, are described in more detail in Chapter 45.

5.1.1.5 Writing Your Own Extensions

If the existing extensions are not sufficient to plug your own study together, the next option is to write your own extension. Again, when writing an extension, one should use the extension points described in Chapter 45, since this is the only way an extension can later become a contribution.

5.1.2 The Ideas Behind this Setup

The setup, as described above, arose from the observation that an-ever growing monolithic MATSim would eventually overwhelm the MATSim team and its core developers group. Therefore, a set-up was sought allowing them to concentrate on central infrastructure, while specific functionality like road pricing, multimodal simulations, signals, additional choice dimensions, or analysis modules could be written and contributed by the community. Clearly, a plug-in architecture had to be the solution, but it took (and still takes) time and effort to make the extension points sufficiently capable and robust.

At the same time, MATSim is a research platform; research investigates innovative questions, which often means that the questions were not foreseen when the code was designed. Quite often, scripting languages are the solution to such problems; for example, python is allowed in QGIS,[4] VISUM (Verkehr In Städten – UMlegung),[5] EMME (Equilibre Multimodal Multimodal Equilibrium), or SUMO (Simulation of Urban Mobility) (via the TraCI interface)[6] for plug-ins. Scala (SCAlable LAnguage) was discussed for MATSim, but ultimately, it was decided to just use Java itself as the scripting language, with the advantage that users between development and MATSim application do not need to learn two languages. In addition, the TU (Technische Universität) Berlin team can continue to teach Java both as an entry point to MATSim and as a general professional skill.

5.2 An Overview of Existing MATSim Functionality

Figure 5.1 shows where common MATSim modules are coupled with the MATSim loop. Some modules have a single connection point (shown around the loop, connected to the respective loop

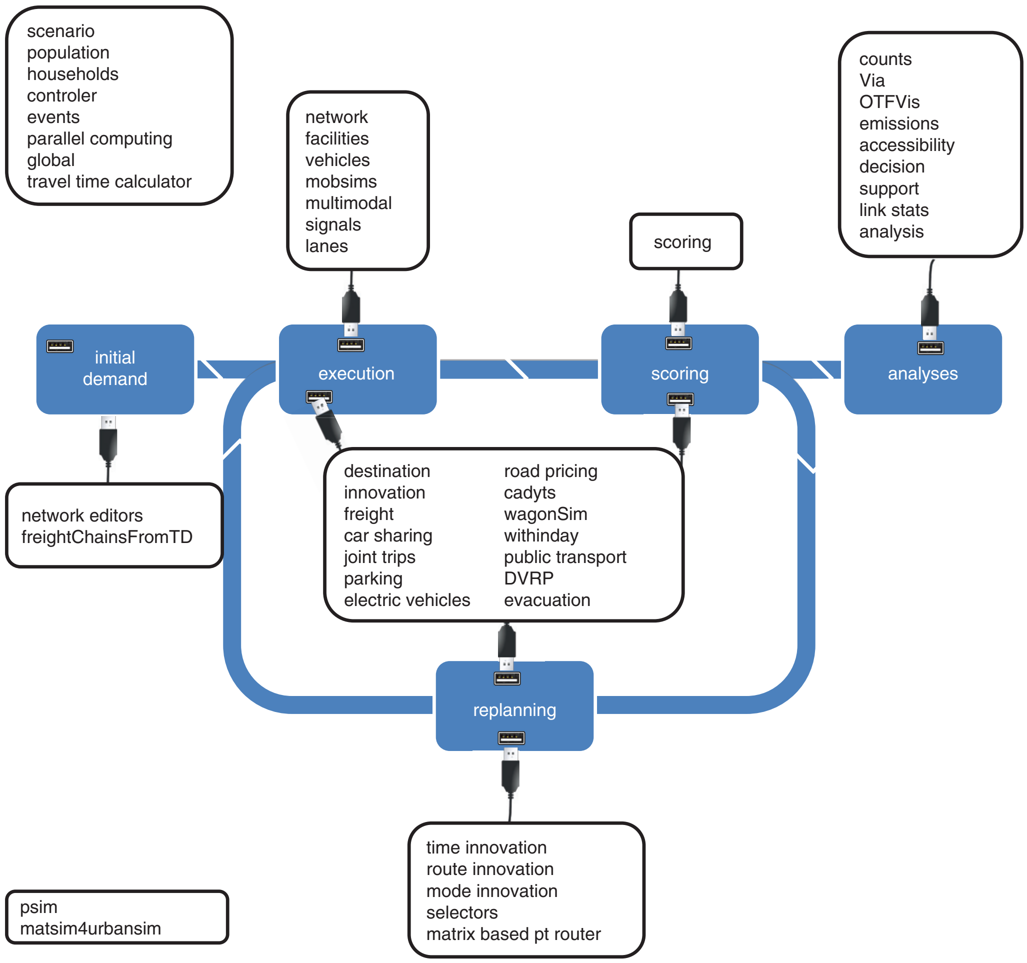

Figure 5.1: MATSim functionality.

[4] `http://docs.qgis.org/testing/en/docs/pyqgis_developer_cookbook/`
[5] PTV (2011)
[6] `http://sumo.dlr.de/wiki/TraCI`

element), while others have multiple connection points (shown in the middle of the circle) and yet others work on a global range (shown on the left upper and lower corners).

The technical details for module usage, in particular, the parameter sets, are described at `http://matsim.org`, especially `http://matsim.org/javadoc` and `http://matsim.org/extensions`.

As a result of the distributed and project- and dissertation-driven MATSim contribution process (see Chapter 44), modules are often implemented for a specific practical purpose, leading to limitations of the respective module. For example, modules might only work for a specific mode, or for a defined calling order. Normally, additional effort is needed to generalize the module; in consequence, the combination of a specific module with other functionality is often not a straightforward task. This means that a user will have to systematically test any specific combination of modules before productively applying it.

The description of the modules in Chapter 4, and the following chapters, is based on the categorization shown in Table 5.1.

Global Modules and Global Aspects	Section 4.2
Controler	Section 4.2.1
Events	Section 4.2.2
Parallel Computing	Section 4.2.3
Global	Section 4.2.4
MATSim Data Containers	Section 4.1 and Chapter 6
Network	Section 4.1.1 and 6.1
Population	Section 4.1.2 and 6.2
Counts	Section 6.3
Facilities	Section 6.4
Households	Section 6.5
Vehicles	Section 6.6
Scenario	Section 6.7
Network Editors	
MATSim JOSM Network Editor	Chapter 8
Map-to-Map Matching Editors in Singapore	Chapter 9
The "Network Editor" Contribution	Chapter 10
Observational Modules	Section 4.7
Travel Time Calculator	Section 4.7.1
Link Stats	Section 4.7.2
Scoring	Section 4.4
Basic Strategy Modules	Section 4.5
Time Innovation	Section 4.5.1.1
Route Innovation	Section 4.5.1.2
Mode Innovation	Section 4.5.1.3
Selectors	Section 4.5.2
Mobsims	
QSim	Section 4.3.1 and Chapter 11
JDEQSim	Section 4.3.2
Individual Car Traffic	
Signals and Lanes	Chapter 12
Parking	Chapter 13

Continued on next page

Electric Vehicles	Chapter 14
Roadpricing	Chapter 15
Other Modes Besides Individual Car	
Public Transport	Chapter 16
The “Minibus” Contribution	Chapter 17
Semi-Automatic Tool for Bus Route Map Matching	Chapter 18
Events-Based Public Transport Router	Chapter 19
matrix-based pt router	Chapter 20
Multi-Modal Contribution	Chapter 21
Car Sharing	Chapter 22
Dynamic Transport Systems	Chapter 23
Commercial Traffic	
Freight Traffic	Chapter 24
wagonSim	Chapter 25
freightChainsFromTravelDiaries	Chapter 26
Additional Choice Dimensions	
Destination Innovation	Chapter 27
Joint Trips and Social Networks	Chapter 28
Socnetgen	Chapter 29
Within-Day Replanning	
Within-day Replanning	Chapter 30
Belief Desire Intention (BDI) Framework	Chapter 31
Automatic Calibration	
Cadyts	Chapter 32
Visualizers	
Via Visualizer	Chapter 33
OTFVis Visualizer	Chapter 34
Analysis	
Accessibility	Chapter 35
Emissions	Chapter 36
Interactive Analysis and Decision Support	Chapter 37
The “analysis” contrib	Chapter 38
Computational Performance Improvements	
PSim	Chapter 39
Other Modules	
Evacuation	Chapter 41
MATSim4UrbanSim	Chapter 42

Table 5.1: MATSim functionality overview.