SUBPART THIRTEEN

Development Process & Own Modules

CHAPTER 44

Organization: Development Process, Code Structure and Contributing to MATSim

Marcel Rieser, Andreas Horni and Kai Nagel

This chapter describes how new functionality enters MATSim. It describes the MATSim team and community, the different roles existing in the MATSim project, the development drivers and processes, and the tools used for integration. The goal is to provide an overview of the development process so that one quickly finds access to the MATSim community and is able to efficiently contribute to MATSim, based on one role or another.

44.1 MATSim's Team, Core Developers Group, and Community

The **MATSim team** currently consists of three research groups and a spin-off company:

- the VSP (VerkehrsSystemPlanung und Verkehrstelematik – The Transport Systems Planning and Transport Telematics group at TU Berlin) group at the ILS (Institut für Land- und Seeverkehr – Institute for Land and Sea Transport Systems), TU Berlin, led by Prof. Dr. Kai Nagel,
- the VPL (VerkehrsPLanung) group at the IVT, ETH Zürich, led by Prof. Dr. Kay W. Axhausen,
- the recently founded Mobility and Transportation Planning group at the FCL, based in Singapore and led by Prof. Dr. Kay W. Axhausen, and
- Senozon AG, based at Zürich with a subsidiary in Germany, founded by former PhD (Philosophiae Doctor – Doctor of Philosophy) and research students.

As is common in research, the university groups' composition changes frequently. Over the last decade more than 50 people, as listed earlier, contributed to MATSim.

How to cite this book chapter:

Rieser, M, Horni, A and Nagel, K. 2016. Organization: Development Process, Code Structure and Contributing to MATSim. In: Horni, A, Nagel, K and Axhausen, K W. (eds.) *The Multi-Agent Transport Simulation MATSim*, Pp. 289–296. London: Ubiquity Press. DOI: http://dx.doi.org/10.5334/baw.44.

A small group of the MATSim team defines the **MATSim core developers group**, maintaining MATSim's core as defined below in Section 44.3.2.

In addition, there is a **MATSim community** composed of closely connected research groups in other cities, e.g., Stockholm, Pretoria, Poznan, and Jülich, as well as more loosely connected external users coming together, e.g., at the annual MATSim User Meeting (see Figure 44.1).

MATSim is open-source software under the GPLv2 (GNU General Public License version 2.0). You are also very welcome to contribute to the code base as described in Section 44.6. New contributors are mentored in the beginning to become familiar with the project and the coding conventions.

44.2 Roles in the MATSim Community

The MATSim community includes the following roles:

- The **MATSim user** uses the official releases or nightly builds and runs the MATSim core with the config file (Section 5.1.1). He or she does not write computer code. Part I of the book is dedicated to the MATSim user. On the web page, he or she finds relevant information in the *user's guide* section and in the user's mailing list `users@matsim.org`.[1] There is also a list of questions and answers under `http://matsim.org/faq`.

  Users should also remember to consult the files `logfileWarningsErrors.log` and `output_config.xml.gz`, as also explained in Section 2.3. The former file is an extract from `logfile.log`, but only contains the warnings and errors. The latter is a complete dump of the currently available configuration options, including comments to most options.
- The **MATSim power user** is a MATSim user with knowledge on how to use the additional modules presented in the book's Part II. He or she does *not* program but knows how to use MATSim scripts-in-Java prepared by others or her/himself, as shown in Section 5.1.1. Parts I and II of the book are helpful to the MATSim power user. Information about extensions can be found under `http://matsim.org/extensions`. Most extensions come with an example script-in-Java. Again, questions and answers are under `http://matsim.org/faq`.
- The **MATSim developer** extends MATSim by programming against the MATSim API (Section 5.1.1). He or she also finds his or her information in Part II of the book, in particular, in Chapter 45, on the web page in the Developer's Guide, and in the mailing list `developers@matsim.org`.
- There are relatively few **MATSim core developers** in the MATSim team. These persons make necessary modifications of the core (as defined in Section 44.3.2), usually after having discussed them in the issue tracker (`http://matsim.org/issuetracker`), in the MATSim committee, or at a developer meeting (see below).

44.3 Code Base

The various pieces of MATSim are delineated by Apache Maven projects and sub-projects. The Apache Maven layout corresponds to the layout of the Git repository.[2] Note that the Java package structure does *not* directly correspond to the Apache Maven/Git layout.

---

[1] During the writing of this book, the information that had so far been contained in the User's Guide was moved to this book. Therefore, the User's Guide section on the web page is currently essentially empty, and may be removed.

[2] MATSim is currently at GitHub under `https://github.com/matsim-org/matsim`. The exact path name may change in the future, e.g., because of changes at GitHub.

Figure 44.1: MATSim events and community.
Source: ©Dr. Marcel Rieser, Senozon AG

44.3.1 Main Distribution

The "MATSim main distribution" corresponds to the "matsim" part of the Git repository. It is the part of the code that the MATSim team feels primarily responsible for. At the time of writing, the MATSim main distribution contains following packages:

- `org.matsim.analysis.*`, containing certain analysis packages that are added by default to every MATSim run.
- `org.matsim.api.*`, see Section 44.3.2.
- `org.matsim.core.*`, see Section 44.3.2.
- `org.matsim.counts.*`, see Section 6.3.
- `org.matsim.facilities.*`, see Section 6.4.
- `org.matsim.households.*`, see Section 6.5.
- `org.matsim.jaxb.*`, containing automatically or semi-automatically generated adapter classes to read XML files using JAXB (Java Architecture for XML Binding).
- `org.matsim.lanes.*`, see Chapter 12.
- `org.matsim.matrices.*`, containing (somewhat ancient) helper classes to deal with matrices, in particular, origin-destination-matrices.
- `org.matsim.population.*`, mostly containing a collection of algorithms that go through the population and modify persons or plans.
- `org.matsim.pt.*`, see Chapter 16.
- `org.matsim.run`, see Section 44.3.2.
- `org.matsim.utils.*` containing various utilities such as the much-used ObjectAttributes (see Section 45.2.2).
- `org.matsim.vehicles.*`, see Section 6.6.
- `org.matsim.vis.*`, containing helper classes to write MATSim information, in particular from the mobsim, to file. This has to a large extent been superseded by the Via visualization package (see Chapter 33).
- `org.matsim.visum.*`, containing code to input data from VISUM.
- `org.matsim.withinday.*`, see Chapter 30.
- `tutorial.*`, containing example code of how to use MATSim, referenced throughout this book.

44.3.2 Core

The core is part of the main distribution (see the previous Section 44.3.1) and contains material that is considered basic and indispensable, and resides in the packages

- `org.matsim.api.*`
- `org.matsim.core.*`
- `org.matsim.run.*`

The MATSim core is maintained by the MATSim Core Developers Group.

44.3.3 Contributions

The idea of the contributions part of the repository is to host community contributions. Historically, most contributors are from the MATSim team, but this is not a requirement.[3] The

[3] It is currently at GitHub under `https://github.com/matsim-org/matsim/tree/master/contribs`.

code is maintained by the corresponding contributor. Code in this section of the repository is considered more stable than code in playgrounds. The Java packages often have the root `org.matsim.contrib.*`, but this is not mandatory.

At the time of writing, there are the following contributions (= extensions which are in the "contrib" part of the repository), listed in alphabetical order:

- `accessibility`, presented in Chapter 35.
- `analysis`, presented in Chapter 38.
- `cadytsIntegration`, presented in Chapter 32.
- `common` is not a true contrib, i.e., it does not provide additional functionality by itself. Instead, it is a place where code used by several contribs, which has not yet made it into the main distribution is located. It also contains some long-running integration tests that are run at each build (i.e., more often than those contained in the `integration` contrib described below).
- `dvrp`, presented in Chapter 23.
- `emissions`, presented in Chapter 36.
- `freight`, presented in Chapter 24.
- `freightChainsFromTravelDiaries`, presented in Chapter 26.
- `grips`, presented in Chapter 41.
- `gtfs2matsimtransitschedule`, presented in Chapter 18.
- `integration` is not a true contrib, i.e., it does not provide additional functionality. Instead, it is a place where integration tests that should run daily or weekly (instead of as often as possible) can be committed.
- `locationchoice`, presented in Chapter 27.
- `matrixbasedptrouter`, presented in Chapter 20.
- `matsim4urbansim`, presented in Chapter 42.
- `minibus`, presented in Chapter 17.
- `multimodal`, presented in Chapter 21.
- `networkEditor`, presented in Chapter 10.
- `otfvis`, presented in Chapter 34.
- `parking`, presented in Chapter 13.
- `roadpricing`, presented in Chapter 15.
- `socnetgen`, presented in Chapter 29.
- `socnetsim`, presented in Chapter 28.
- `transEnergySim`, presented in Chapter 14.
- `wagonSim`, presented in Chapter 25.

44.3.4 Playgrounds

Another element of the MATSim repository is the "playgrounds". These are meant as a service to programmers. They have grown historically from the fact that MATSim's object classes and in consequence the interfaces between them have evolved and grown over time, and thus a stable API was not available. Regular code-wide refactorings, along the lines discussed, e.g., by Fowler (2004), were thus the norm for many years.

At this point, the extension points described in Chapter 45 should be somewhat stable and development against them should be possible without major changes from release to release. Anybody who needs tighter integration with the project should still apply for a playground.

44.3.5 Contributions and Extensions

Congruent with the structure of this book, the MATSim code structure contains a core which allows to run basic MATSim using the config file, a population and a network. Packages going beyond this basic functionality are extensions, where three different kind of extensions exist:

- extensions in the main distribution,[4]
- extensions contributed by the MATSim community known as contributions, and
- any code written anywhere published or unpublished extending the MATSim core.

Extensions are listed at `http://matsim.org/extensions`.

44.3.6 Releases, Nightly Builds and Code HEAD

Releases, nightly builds and the code head can be obtained from `http://matsim.org/downloads`.

MATSim releases are published approximately annually. Usually, MATSim users and MATSim power users as defined above in Section 44.2 work with releases.

MATSim uses continuous integration and, thus, nightly builds are available without stability guarantee under `http://matsim.org/downloads/nightly`. MATSim API developers that depend on a very recent feature might use Nightly builds.

Both Apache Maven releases and Apache Maven snapshots are available, see `http://matsim.org/downloads` for details.

MATSim API developers or core developers often work on the code's HEAD version that can be checked out from GitHub.

Nightly builds and maven snapshots are only generated when the code compiles and passes the regression tests. They are, in consequence, somewhat "safer" than the direct download from the HEAD.

44.4 Drivers, Organization and Tools of Development

Important drivers of the MATSim development are the projects and dissertations of the MATSim team. New features are developed as an answer to requirements of these dissertations and projects, where projects range from purely scientific ones—often sponsored by SNF (Schweizerischer Nationalfonds) or DFG (Deutsche Forschungsgemeinschaft)—via projects for governmental entities and projects where science and industry contribute equally—e.g., CTI (Commission for Technology and Innovation) projects—to purely commercial projects, which are managed by Senozon AG in the majority of cases. A significant number of innovations are also introduced by the collaboration with external researchers.

Systematic code integration is mainly performed by the Berlin group and by Senozon AG. This includes continuous code review and integration upon request of the community, but also comprehensive code refactorings to clean up code and to improve modularity. Refactorings are discussed and documented in the MATSim issue tracker (`http://matsim.org/issuetracker`).

The development process is supported by a MATSim standing committee discussing software and sometimes conceptual issues on a regular basis (`http://matsim.org/committee`). Another element that brings in innovation as well as organization are the annual meetings. Right from the beginning, there have been a MATSim developer meetings focused on coding issues. Later, a user meeting offering insights into current work by the community has been added, sometimes

[4] At the time of writing it is unclear if these extensions might one day become contributions, shrinking the MATSim main distribution to its core.

combined with a tutorial. Finally, a conceptual meeting is now held every year, concentrating on issues that go beyond pure software engineering. The developer meeting and the conceptual meeting together establish the road map that guides development for the remainder of the year.

MATSim development makes use of a large number of tools, hopefully leading to better software quality. Historically, many of those tools ran from automated scripts and were made available at `http://matsim.org/developer`. Nowadays, most of them are automatically available from the build server (see `http://matsim.org/buildserver`) and/or from the repository (`https://github.com/matsim-org/matsim`), so that many of them are scheduled for removal from `http://matsim.org/developer`. Some of these tools are: a change log; an issue tracker; the javadoc documentation; static code analyses performed by *FindBugs* and *PMD*; test code coverage analysis; copy paste analysis; code metrics; Apache Maven dependencies; and information about the nightly test results. These nightly test results are generated by the MATSim build server based on the Jenkins software.

Furthermore, there is a MATSim benchmark at `http://matsim.org/files/benchmark/benchmark.zip`. For results see `http://matsim.org/benchmark`.

Most MATSim developers use Eclipse as an IDE. The MATSim documentation is tailored to this IDE. Team development is currently based on Git as revision control system. External library dependencies are managed by Apache Maven.

44.5 Documentation, Dissemination and Support

The main documentation is now this book. Additional information, including tutorials, can be found under `http://matsim.org/docs`. Code documentation in form of javadoc can be found unter `http://matsim.org/javadoc`.

For fast application of MATSim, some small-scale example scenarios are provided in the code base (`folder: examples`), where recently an extended version of the well-known benchmark scenario for the City of Sioux Falls has been added (Chakirov and Fourie, 2014) (Chapter 59). Additional example datasets, including Berlin datasets, can be obtained via `http://matsim.org/datasets`.

Further information is disseminated at the afore-described annual user meetings and MATSim mailing lists, see `http://matsim.org/mailinglists`. Support is provided by the MATSim team via these mailing lists and via `http://matsim.org/faq`, both on a best effort basis. Many components of MATSim are documented by the numerous papers published in international journals and presented at worldwide conferences. Information about such publications can, e.g., be obtained from `http://matsim.org/publications` and from this book.

44.6 Your Contribution to MATSim

The technical details, i.e., the MATSim extension points, on where to hook with MATSim are detailed in Chapter 45. Here, the different ways of contributing to MATSim according to the roles presented in Section 44.2 are introduced.

As a MATSim user, power user, or API developer, you are warmly welcome to make an important impact by reporting your achievements, needs and problems with, or bugs of, the software via the users mailing list, the issue tracker, the FAQ, or at the annual MATSim user meeting.

If you would like to directly contribute to the code base of MATSim, you are welcome to become part of the contributions repository.

If you are the type of person that likes to change the core system, you can, although it is a long way, become a member of the MATSim core developers group. Core developers are usually picked from the MATSim team. Prerequisites are a strong computer scientist background, several years of experience with MATSim and a deep understanding of large software projects.