STATUS OF JMAD, THE JAVA-API FOR MADX

K. Fuchsberger, X. Buffat, Y.I. Levinsen, G. J. Müller (CERN, Geneva, Switzerland)

Abstract

MadX (Methodical Accelerator Design) is the de-facto standard software for modeling accelerator lattices at CERN. This feature-rich software package is implemented and still maintained in the programming languages C and FORTRAN. Nevertheless the controls environment of modern accelerators at CERN, e.g. of the LHC, is dominated by Java applications. A lot of these applications, for example, for lattice measurement and fitting, require a close interaction with the numerical models, which are all defined by the use of the proprietary MadX scripting language. To close this gap an API to MadX for the Java programming language (JMad) was developed. JMad was first presented to the public about one year ago. In the meantime, a number of improvements were done, and additional MadX features (e.g., tracking) were made available for Java applications. Additionally, the graphical user interface was improved and JMad was released as open source software. This paper describes the current status and some new features of the project, as well as some usage examples.

INTRODUCTION

MadX [1] is the latest iteration of the highly successful MAD program code, used by a very large community at CERN and elsewhere. Numerous lattice models exist for most accelerators at CERN, including the SPS, the LHC and the transfer lines in between which are regularly maintained and updated. MadX is designed as a standalone software with its own proprietary scripting language, which is used to interface with the software. Nevertheless, in many situations (e.g. postprocessing of data, plotting, fitting) it is more appropriate to access MadX from other higher level languages, for which a well defined API (Application Programming Interface) is essential. Since this was already outlined in more detail in [2], we will not repeat the full discussion here.

JMad offers such an API for the Java programming language. Internally it communicates with MadX through pipes and files and provides the user with plain java objects to change the status of the model and retrieve results. The necessary MadX binaries are packaged inside the library and are extracted when needed. This architecture allows to use the JMad API wherever any other java library could be used, provided that a MadX executable is available for the operating system in use (which is currently true for Linux, OSX and Windows).

As illustrated in Fig. 1, the most important components of the JMad architecture are:

- **JMad Service:** This is the main facade component for an application which is using the API. The key responsibility of this interface is to find available model definitions and create model instances from these definitions.
- **JMad Model:** This is the key component of JMad. It is represented by the `JMadModel` interface. Each `JMadModel` instance is associated to one dedicated MadX process. The `JMadModel` interface provides Java methods to act on the model (e.g. run a twiss calculation, get/set strength values and many more) which are passed on to the MadX process.
- **Model Definitions:** A model definition contains all the information which is necessary to initialize the MadX process (e.g. all the required sequence- and strength-files) as well as available options which are possibly selectable by the user/application (like available sequences, ranges or optics). In the API, a model definition is represented by the `JMadModelDefinition` interface. Model definitions can be either imported from files or can be contained directly in java libraries. For the latter one, JMad contains an auto-detection mechanism. This mechanism searches for model definitions contained in the Java class path in a distinct package.

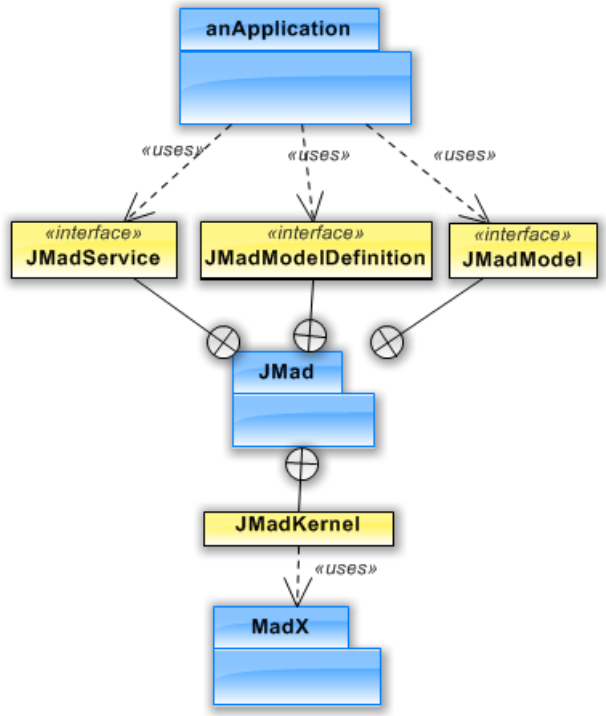

Figure 1: The JMad key components.

RELEASE AS OPEN SOURCE

JMad was first presented on last years IPAC conference. There was a lot of positive response from many interested MadX users, both from CERN and other institutes. Because of the high interest from outside CERN, the main goal was to open this tool to a broader community. Finally, this could now be achieved, just before the publication of this paper:

A binary version, installation instructions and the whole documentation are available on the JMad website[1]. The sourcecode of JMad was released as open source under the Apache 2.0 License[2] and is available on Github[3].

NEW FEATURES

Next to the release of JMad as open source, some interesting new features were added during the last year. The most important ones of them are:

- Exposure of the import/export feature for model definitions to the graphical user interface: The user now can directly save model definitions to zip files or to plain files. Using this feature now makes it easy to create new model definitions and exchange them between the users.
- Exposure of part of the MadX tracking functionality to the API: The 'dynap'-functionality was exposed to the Java side. This allows to create tune-footprints and will be used to study beam-beam effects in the LHC.

JMAD GRAPHICAL USER INTERFACE

Although the intended purpose of JMad is to be used as a library, also a graphical user interface (GUI) was created. This GUI allows to open different models, change arbitrary settings of the models and create plots of the resulting optics functions. Since it has undergone major changes since the first release, it deserves some detailed description as given in the following:

Figure 2 shows a screenshot of the main frame of the JMad GUI. The numbered areas in this frame have the following purpose:

1. **Menus and toolbar:** The toolbar provides buttons to open a new model and to choose different ranges and optics for the actual model. Also the import and export of model definitions (to/from zip files and plain files) is supported in the latest version.
2. **List of models:** All the actually opened models are displayed in this list. In the example, two models ('longti2' and 'LHC (LSA)') are opened. By selecting one of the models in the list, the selected model becomes the active one and its values are displayed and can be edited in the other areas.
3. **Model-operations panel:** In this area, properties of the model can be changed. Editable values are for example twiss initial conditions, strength values or properties of individual elements. In the example, the panel for changing individual element-properties is shown: On the left, a list of all available elements is displayed. The right half of the panel contains an editable table of all the properties of the selected element. The optics values at the position of the selected element are listed in a separate table.
4. **Data panel:** Selected output data is shown in this panel. In the example a tune diagram is displayed.
5. **Dataviewer-explorer:** In this area, all the currently available plots are listed and can be selected.
6. **Dataviewer:** In this area, the plots that are selected in the Dataviewer-explorer, are displayed. The example shows a plot of the beta- and dispersion-functions of the LHC around IP5, using two different y-axes. The example plot also demonstrates the possibility to mark certain elements in the plots (e.g. the position of the interaction point IP5 is shown in the plot, simply by selecting the check box 'mark' in the elements-table of area (3)).
7. **Plot buttons:** Below the data view two buttons are displayed: The button 'Add view' opens a separate dialog in which a new plot can be defined. The button 'Refresh all views' recalculates the optics values and updates all plots and the output data in the data panel (4). The button '>>> ref' copies the current optics values to a reference dataset, that can be used later on to produce difference plots.

APPLICATIONS

The most important application using the JMad package is the Optics and Knob Management application for the LHC, which is part of the LHC online model project [4, 5]. This application allows the semi-automatic processing and transfer of huge amounts of optic data input and high-level manupulation parameter definitions (knobs) into the LHC control system. The usage of JMad was a natural choice for this project since it provides a framework to nicely define and manage optic models as well as file-indepenent interaction with MadX. The tools provided for operation contributed to a great extends to the smooth commissioning of the "90 m β* optics" [6] as well as the "Achromatic Telescopic Squeezing (ATS) Scheme" [7]. Including the nominal LHC optics, a total of 57 optics including 2040 knobs have been uploaded to the control system and are used for operation

JMad is also an essential ingredient for the LHC Aperture Meter [4, 5]. This application, available in the control room, retrieves the actually measured orbit at the beam position monitors and calculates interpolated values for all elements of the LHC, taking into account the active optics in the LHC. JMad provides the simulation input for these calculations. Combined with the information from the the-

[1] http://cern.ch/jmad
[2] http://www.apache.org/licenses
[3] https://github.com/jmad/jmad

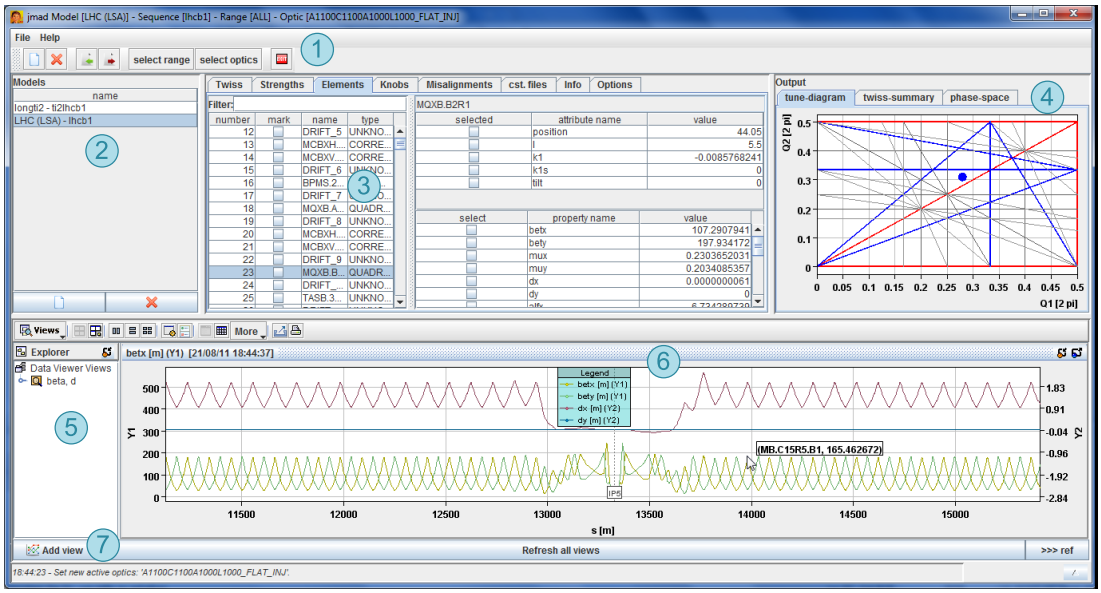

Figure 2: Overview of the JMad GUI.

oretical aperture model and measured positions of movable devices in the vacuum pipes, the available aperture is calculated and warnings are displayed if the available free space drops below certain limits.

JMad functionality was also heavily used when optimizing the LHC squeeze beam processes: The impact of the interpolation of the magnet currents between matched points in the beam process was simulated using JMad and corrections were calculated which were then fed forward to the machine-settings in order to minimize the errors [8].

Aloha (*Another Linear Optics Helper Application*), the project which originally triggered the development of JMad is still a very important application. It uses JMad to calculate response matrices in order to produce model fits to measured data [9]. This technique was used extensively during the commissioning of the LHC transfer lines [10].

INTEGRATION WITH PYTHON

At the same time as JMad became an open source project, an API for MadX for the Python language (PyMad) was released [3]. JMad itself is an important part of this. The combination of JMad and PyMad allows to interact with the models through a graphical user interface and alternatively run Python scripts to retrieve data from the models or change model settings.

SUMMARY AND OUTLOOK

The main achievement for JMad over the last year was the release as open source project. This also allows users outside of CERN to use it and even make changes and contribute to the improvement of the library.

Next plans are to include a simple plugin system in the JMad GUI, which then allows to e.g. add optional online functionality to JMad. Further, the development of GUI components to display tune footprints is in preparation.

ACKNOWLEDGEMENTS

Many thanks T. Baer, J. Wenninger, R. Schmidt and V. Kain for all their feedback. Special thanks to R. Gorbonosov an V. Baggiolini for all the software design related discussion and to N. Stapley and K. Sigerud for all their help and feedback concerning the licensing of JMad and releasing it as open source.

REFERENCES

[1] W. Herr, F. Schmidt “A MAD-X Primer”, CERN AB Note, CERN-AB-2004-027-ABP.

[2] K. Fuchsberger et al., “JMad - Integration of MadX into the Java World”, proceedings IPAC 2010.

[3] K. Fuchsberger and Y.I. Levinsen, “PyMad - Integration of Mad-X in Python”, these proceedings.

[4] G. Müller et al., “The Online Model for the Large Hadron Collider”, proceedings IPAC 2010.

[5] G.Müller, “LHC Online Modeling”, Optics Measurements, Corrections and Modeling for High-Performance Storage Rings Workshop, CERN 2011.

[6] S. Cavalier et al., “Commissioning of the 90 m Optics”, these proceedings.

[7] S. Fartoukh, “An Achromatic Telescopic Squeezing (ATS) Scheme for the LHC Upgrade”, these proceedings.

[8] X. Buffat et al., “Simulation of linear beam parameters to minimize the duration of the squeeze at the LHC”, these proceedings.

[9] K. Fuchsberger, “Aloha - Optics studies by combined kick-response and dispersion fits”, CERN BE-Note-2009-020 OP.

[10] K. Fuchsberger, “Novel Concepts for optimization of the CERN Large Hadron Collider Tranfer Lines”, phd thesis, CERN 2011.