developer.skatelescope.org Documentation

Release 0.1.0-beta

Marco Bartolini

Oct 12, 2022

Home

This repository tracks the status of the development of a UML/VODML model for the Measurement version 3 with modifications to address shortcomings or special SKA requirements. The base MSv3 model, which is the VO-DML compliant model of the MSv3 as described in the document with some modifications to address modelling issues, will be kept as a permanent release tag, all additional modifications will then happen on a SKA branch.

The model has been created using the Modelio 4.1.0 UML modeller (public version). It is available for free on various platforms and can be downloaded from the Modelio web site. Once installed the SKA MSv3 model can be loaded directly from the repository by pointing the Modelio workspace path to the Modelio/workspace folder in the repository.

Alternatively this repo also contains a Dockerfile in the Modelio subfolder to generate a docker container which will mount the workspace from this repo directly when started. This requires to download the Modelio debian package from

```
https://www.modelio.org/download.html?task=download.send&id=171&catid=37&m=0
```

and then run:

```
docker build -t modelio:latest ./
```

starting the resulting container is quite complex, because it requires setting up a X connection to the host and mount the repo folder. In order to facilitate that you can execute the shell script

```
./modelio.sh
```

CHAPTER 1

VODML toolchain

One of the goals of this modelling exercise is to arrive at a proper data model, which also verifies as a valid IVOA data model using the VO-DML toolchain. This is described in serveral documents and also a tutorial linked from the main VO-DML wiki page:

```
https://wiki.ivoa.net/twiki/bin/view/IVOA/VODML
```

In order to adapt this we have modified the SKA-MSv3 model in a way that it is running through the translation and the verification. After that it is also possible to generate a set of HTML pages containing the descirption of the model as extracted from the model. At this stage the latter is not very useful, because the amount of descriptive detail in the model is not sufficient. The VO-DML toolchain consists of a volute repository and requires a complete Java development environment, including ant build environment to be available. Forking the vo-dml repository into the SKA would defeat the purpose of staying in-line with the IVOA effort, thus we are taking the approach of using the toolchain directly from the original source. The repository can be downloaded using subversion:

```
cd /tmp
svn checkout https://volute.g-vo.org/svn/trunk/projects/dm/vo-dml
```

The vo-dml repository also contains a set of documents, template data models and other models. In order to run the SKA-MSv3 through the chain the model has to be exported from Modelio into an XMI document and copied in a new folder in the cloned vo-dml folder. This is faciliated by a directory contained in this repository which has the same sub-structure as the vo-dml repository one and already contains an exported XMI file.

```
./vo-dml/models/SKA-MSv3/vo-dml
```

rsync the two structures:

```
rsync -avz ./vo-dml/ /tmp/vo-dml/
```

and then run the translation into VO_DML:

```
cd /tmp/vo-dml/tools
ant run_xmi2vo-dml
```

There will be plenty of output on the console. After that you can run the verification:

```
ant run_validate_vo-dml
```

This will generate a whole set of warnings in a log file, since the current model is using complex types and VO-DML does not support that at this point. Ignoring that for now you can still run the HTML generation as well. This requires that the graphviz package is installed and the graphviz.path property is correct in the file vo-dml/tools/build.properties (it is correct for a MAC OSX installation of graphviz using brew). To generate and display the HTML run

```
ant run_vo-dml2html
```

on a MAC you can open the HTML page using:

```
open ../models/SKA-MSv3/vo-dml/SKA-MSv3-0.1.html
```

Clicking on the first link (*model: SKA_MeasurementSet*) will take you to a graphical representation of the model with all model elements.

CHAPTER 2

Model description

The SKA-MSv3 has been modelled in a bottom-up approach, starting with a proper modelling of a MS table. Such a model does not exist and is not described anywhere else. Without such a base model quite a number of details of the MS system are implicit and left to interpretation. In order to prevent that from happening the SKA version defines the MS table in a ridgid way. The MSv3 table model is a column based model and thus the most basic class in this whole model is the

```
MS_Table.MsTableColumn_class
```

this class is defining all mandatory and optional attributes required to properly define a MSv3 table column. In this model this is used as a complex type definition, which is prefectly valid in UML, but not in VO-DML (and thus the warnings mentioned above). When modelling the actual tables each column is thus of type MsTableColumn_class.

This also points to a quite fundamental property of the MSv3 table model: It actually is a

meta-model

which means that an instance of this model is a set of classes rather than an object.

In principle this is only required for the basic MS table definition, but since it has not been done before we have started at that level and it is included here. Every actual implementation of a MSv3 table (for example the MAIN table class) is then an instance of this meta-model and thus a class, but obviously not yet an actual MAIN table of a physical MeasurementSet. Instantiating such a table class needs to happen when generating a physical MeasurementSet. Meta-modelling is quite tricky even in UML, since the meta-model is actually implemented on the tool level rather than the model level. If you switch Modelio (or any other UML tool) from modelling UML to modelling and Entity-Relationship model you are changing the meta-model of the tool itself, which gives you access to the definitions of the meta-model. The current model is more or less just a short-cut and the next step would be to implement a proper meta-model for Modelio (for instance). Unfortuntely there is no standard for this kind of modelling and it would thus lock us to a certain tool.

Once done, the beauty is that you could simply switch the tool to *MS modelling* and the only options you would see are proper MS table and column classes, rather than generic classes. When creating a new model instance, e.g. the SKA-MSv3, you would start with all the MSv3 mandatory tables and the option to add additional columns and tables.

In addition to these fundamental model properties there is also a lot of (hidden) detail in the model. One of the weaknesses of the current MSv3 draft is the fact that there is no top level MS entity. The model in this repo has

a proper entity, which encapsulates all the required and optional tables. At the instance level this would also be an actual entity of some sort, which binds everything together, by back-referencing to its unique ID. All entites in this model have unique IDs and are referencing back to the top-level MS entity. That means that any table object 'knows' which MeasurementSet it belongs to or, even more importantly, every column object 'knows' which table it belongs to, regardless of the physical implementation of the model.

Another hidden feature of this model is the inclusion of the ASDM enumerations. For instance in ASDM almost none of the values in any table column are defined as just strings, but are of type enumeration and all of them are properly defined. This means in turn that the actual values can be verified against the model and that makes the whole system a lot more reliable. These enumerations have not been mapped into the MSv3 yet, but we will do this as part of the SKA-MSv3 instantiation.

Todo:

- Insert todo's here

CHAPTER 3

Package-name documentation

This section describes requirements and guidelines.

3.1 Subtitle

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

3.1.1 Public API Documentation

Functions

Classes

CHAPTER 4

Project-name documentation HEADING

These are all the packages, functions and scripts that form part of the project.

- *Package-name documentation*