International Virtual Observatory Alliance

Implementation of the Simulation Data Model for theoretical databases

Version 1.0

IVOA Note 2010 November 26th

This version: 1.0
Version 1.0-20101126

Latest version:
http://www.ivoa.net/Documents/latest/latest-version-name

Previous version(s):

Editors:
Franck Le Petit, Benjamin Ooghe-Tabanou

Author(s):
Franck Le Petit, Benjamin Ooghe-Tabanou, Nicolas Moreau, Laurent Bourgès, Jonathan Normand

Abstract

This document aims at providing examples of implementation of SimDM / SimDB. Two VO-Theory data bases are used as example. First, the STARFORMAT database, that publishes MHD simulations of the interstellar gas. Second, the PDR database, that publishes 1D models of interstellar clouds. We present how the results of these simulations have been mapped on the Simulation datamodel. We also present in each case, the pipeline we used to simplify the updates of the databases each time the codes behind the simulation evolves or that new simulations have to be added in the databases.

Status of This Document

This is an IVOA Note expressing suggestions from and opinions of the authors. It is intended to share best practices, possible approaches, or other perspectives on interoperability with the Virtual Observatory. It should not be referenced or otherwise interpreted as a standard specification.
A list of current IVOA Recommendations and other technical documents can be found at http://www.ivoa.net/Documents/.

Acknowledgements

We would like to thanks Gerard Lemson, Fabrice Roy, Hervé Wozniak for the VO-Theory I.G. and Mireille Louys from the Data Model WG for their helpful comments.

Contents

1 Introduction 4
2 Starformat – MHD simulations 5
2.1 Description and objectives 5
2.2 SimDM implementation on STARFORMAT 5
2.2.1 - Project part 5
2.2.2 Protocol part 6
2.2.3 Experiment part 6
2.2.4 Snapshot part 7
2.2.5 PostProcessor & PostProcessing part 8
2.3 Implementation SimDB / VO-URP 9
2.3.1 VO-URP environment 9
2.3.2 Ingestion pipeline 9
2.4 Comments 10
3 PDRDB – Micro-physics simulations 12
3.1 Description and objectives 12
3.2 SimDM implementation on PDRDB 13
3.2.1 - Protocol part 13
3.2.2 Experiment 14
3.2.3 Object 15
3.2.4 - Projects 16
3.3 Implementation SimDB / VO-URP 17
3.3.1 - Database 18
3.3.2 - Ingestion of data in PDRDB 18
3.3.3 Services 20
Appendix A: SimDB Classes Instantiation 21
References 22

1 Introduction

The simulation data model (SimDM) has been developed by the VO-Theory Interest Group at IVOA. Initially, one of the major objectives was to publish 3D + time simulations. Quite quickly the Data Model (now known as SimDM) evolved to cover larger sets of simulations. Because simulations and numerical models can be very heterogenous, the drawback is that SimDM may be one of the most difficult data model to implement.
In this note, we present two examples of implementation of SimDM: one about 3D+times MHD simulations, and one about 1D stationary models of the atomic and molecular structure of interstellar clouds. These two examples cover two extreme cases for the implementation of SimDM.

Among the major steps one has to do to develop such VO-Theory services, we recommend to the developers to pay attention to two points at the beginning of the project :

1) The first question any publisher should ask to himself, is what do I want to publish and for who ? Simulations can be quite complex and may provide many heterogeneous quantities. Is it really useful to publish all the computed quantities ? In the two services presented here, we decided to publish in the database only the main quantities to allow users to find the simulations that may interest them. Indeed, this is the aim of SimDM, to help users to discover simulations, not to publish all the results of simulations in a SimDB. Quantities not published in SimDB can still be accessible by a download service. For example, in the PDR service, we published quantities as line intensities and column densities that can be used by observers to search interesting models. But we did not publish in SimDB local emissivities or cooling and heating rates by various physical processes. Nevertheless, these quantities, not published on the database, can be obtained by a download of the full result.

2) The other aspect we would like to highlight is that the developer has to think at the beginning of the project to the future evolutions of the SimDB service. Codes can be modified frequently to add new physics. As a consequence, the quantities published in a VO-Theory service may have to be modified frequently. Obviously, we wish this be as transparent as possible for the scientist publishing his data. To solve this issue, we developed a pipeline to ingest simulations / models in the data bases. For example, in the case of the PDR code, outputs of the code directly contains an XML file describing the content of the outputs. The pipeline, that makes use of VO-URP, analyzes this XML and if it identifies new quantities it can modify the database automatically.

2 Starformat – MHD simulations

2.1 Description and objectives

StarFormat stands for Star Formation. It is a grouped database assembling multiple simulation models from the results of theoretical projects designed to study the formation of molecular clouds and prestellar dense cores in order to aid in the analysis and interpretation of current and future observations. It contains so far 4 different sets of large MHD simulations using the three-dimensional adaptive mesh refinement (AMR) magneto-hydrodynamical code RAMSES-MHD. Tryouts have been performed and StarFormat will soon host also very different kinds of simulation like turbulence boxes or dense cores from projects using the other popular codes FLASH, GADGET or PHOENIX.

The service proposes a synthesized access to all simulations descriptions, parameters and snapshots. Post-processings applied to these snapshots in order to identify and characterize specific objects like dense clumps are described extensively and can be queried along the values of a selection of properties. Data cubes extraction from the raw data is proposed as an extra service. TAP access to the content of the whole database will also be provided.

2.2 SimDM implementation on STARFORMAT

The SimDM data model provides a rich and exhaustive representation of all the elements needed to describe extensively the StarFormat simulations.

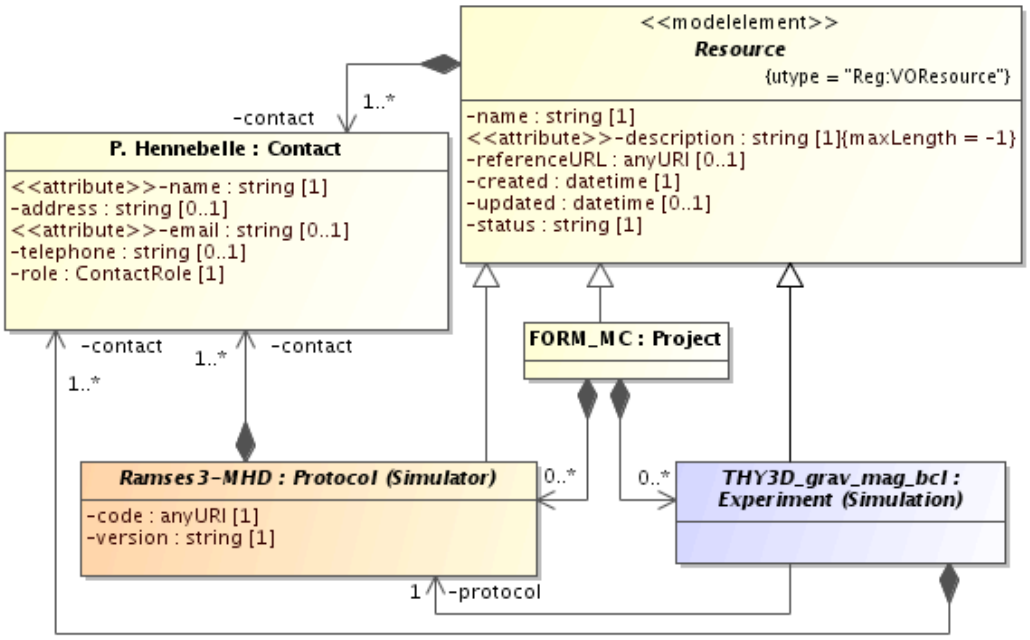
2.2.1 - Project part

uType: SimDB:simdb/Project

All the simulations are stored and described within different instances of the Resource class: different versions of the RAMSES codes and their runs are respectively stored as instances of the classes Protocol and Experiment. These are grouped as ProjectResources within a “Formation of Molecular Cloud” Project, third kind of possible Resource. Any participant to the work can be associated with his complete coordinates to these different levels of Resource as a Contact assuming different possible roles such as owner, creator, publisher or contributor.

uTypes:

- *Resource: SimDB:simdb/Resource*
- *Project: SimDB:simdb/Project*
- *ProjectResource: SimDB:simdb/ProjectResource*
- *Contact: SimDB:simdb/Contact*



Example of StarFormat instances for the Project part

2.2.2 Protocol part

uType: SimDB:simdb/protocol/Protocol

The RAMSES code used is described within the Simulator class, an instance of the Protocol class. Using The Simulator's various heritage classes, the code is fully described with its different components: all the possible physical processes which can be part of a run (MHD, hydrodynamics, gravitation, heating and cooling, ...) are described within the Physics class; the parameters (AMR levels, boundary and initial conditions, incoming flow velocity, ...) are stored as InputParameters and grouped within a ParameterGroup as ParameterGroupMembers whenever they come as a series of values like for example the jeans refinement lengths. Finally the use of adaptive mesh refinement is specified within the RepresentationObjectType class in which we describe the basic AMR cells that compose the data cube for a RAMSES simulation.

uTypes:

- *Protocol: SimDB:simdb/protocol/Protocol*
- *Simulator: SimDB:simdb/protocol/Simulator*
- *Physics: SimDB:simdb/protocol/Physics*
- *InputParameter: SimDB:simdb/protocol/InputParameter*
- *ParameterGroup: SimDB:simdb/protocol/ParameterGroup*
- *ParameterGroupMember: SimDB:simdb/protocol/ParameterGroupMember*
- *RepresentationObjectType: SimDB:simdb/protocol/RepresentationObjectType*

2.2.3 Experiment part

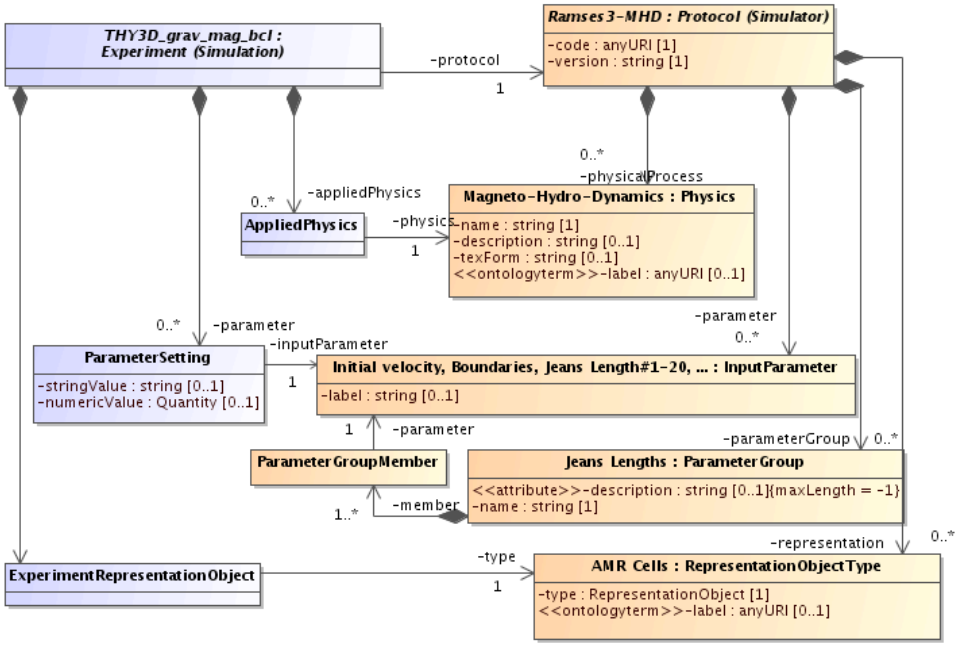
uType: SimDB:simdb/experiment/Experiment

Any run of the code is then described as a Simulation, which is an instance of Experiment class, and refers to the corresponding Protocol of class Simulator. The specificities of each simulation are

described though the uses of the corresponding Experiment's heritage classes referring to the Protocol's corresponding ones: the physical processes applied are notified through instances of AppliedPhysics as references to the Physics elements defined earlier; all the parameters of the run, whether they are numerical values or textual, are given as instances of string or quantity ParameterSetting referring to the previous InputParameter class. The molecular cloud simulated by a run is described as a TargetObjectType that can be characterized by properties further on. An instance of ExperimentRepresentationObject pointing to the previous RepresentationObjectType indicates the use of AMR representation for the simulation and ExperimentProperty instances are used to point to the properties characterizing these representation like the number of cells.

uTypes:

- *Experiment: SimDB:simdb/experiment/Experiment*
- *Simulation: SimDB:simdb/experiment/Simulation*
- *AppliedPhysics: SimDB:simdb/experiment/AppliedPhysics*
- *ParameterSetting: SimDB:simdb/experiment/ParameterSetting*
- *TargetObjectType: SimDB:simdb/experiment/TargetObjectType*
- *ExperimentRepresentationObject: SimDB:simdb/experiment/ExperimentRepresentationObject*
- *RepresentationObjectType: SimDB:simdb/experiment/RepresentationObjectType*



Example of StarFormat instances for the Protocol and Experiment parts

2.2.4 Snapshot part

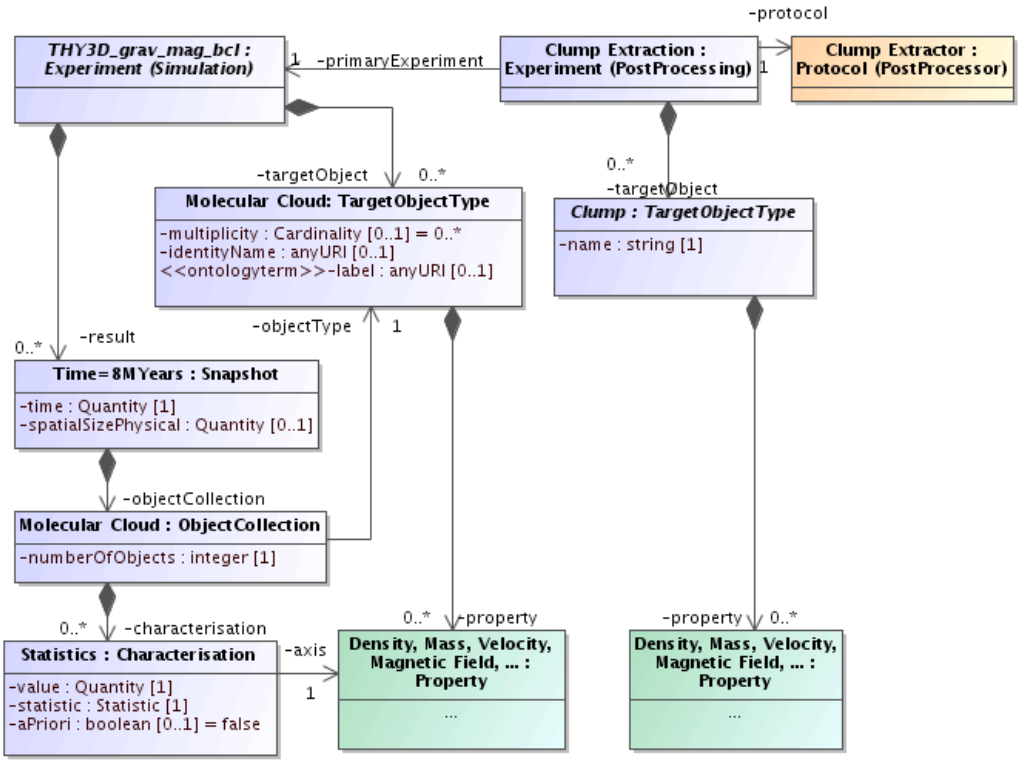
uType: SimDB:simdb/experiment/Snapshot

Each timestep of the simulation is stored as a Snapshot. For each snapshot, ObjectCollection instances referring to the cloud and the AMR (i.e the run's representation and target objects) are associated and characterized though the use of Statistics instances providing values for the Object's Property classes.

uTypes:

- *Snapshot: SimDB:simdb/experiment/Experiment*

- *ObjectCollection: SimDB:simdb/experiment/ObjectCollection*
- *Statistics: SimDB:simdb/experiment/Statistics*
- *Property: SimDB:simdb/object/Property*



Example of StarFormat instances for the Snapshot and PostProcessing parts

2.2.5 PostProcessor & PostProcessing part

uTypes: SimDB:simdb/protocol/PostProcessor & SimDB:simdb/experiment/PostProcessing

Finally the postprocessing applied to each snapshot in order to extract dense cores out of the molecular cloud is saved as a PostProcessor, another instance of the Protocol class referring to its parent Simulator and which can be described extensively in the same ways as the parent Protocol. Each Snapshot of the previous Simulations has multiple applications of PostProcessings referring to this PostProcessor child protocol and are referring to their parent Experiment as well as to the corresponding Snapshot through the use of the InputDataSet class.

These PostProcessings, or child Experiments, have as many Snapshot as extracted dense cores of matter, qualified as “clumps”. The clumps are described as TargetObjects of the PostProcessing and each individual one is saved as a different Snapshot of time 0 for the PostProcessing Experiment. They can therefore be characterized in identical ways as the cloud for the parent Experiment.

uTypes:

- *PostProcessor: SimDB:simdb/protocol/PostProcessor*
- *PostProcessing: SimDB:simdb/experiment/PostProcessing*
- *InputDataSet: SimDB:simdb/experiment/InputDataSet*

2.3 Implementation SimDB / VO-URP

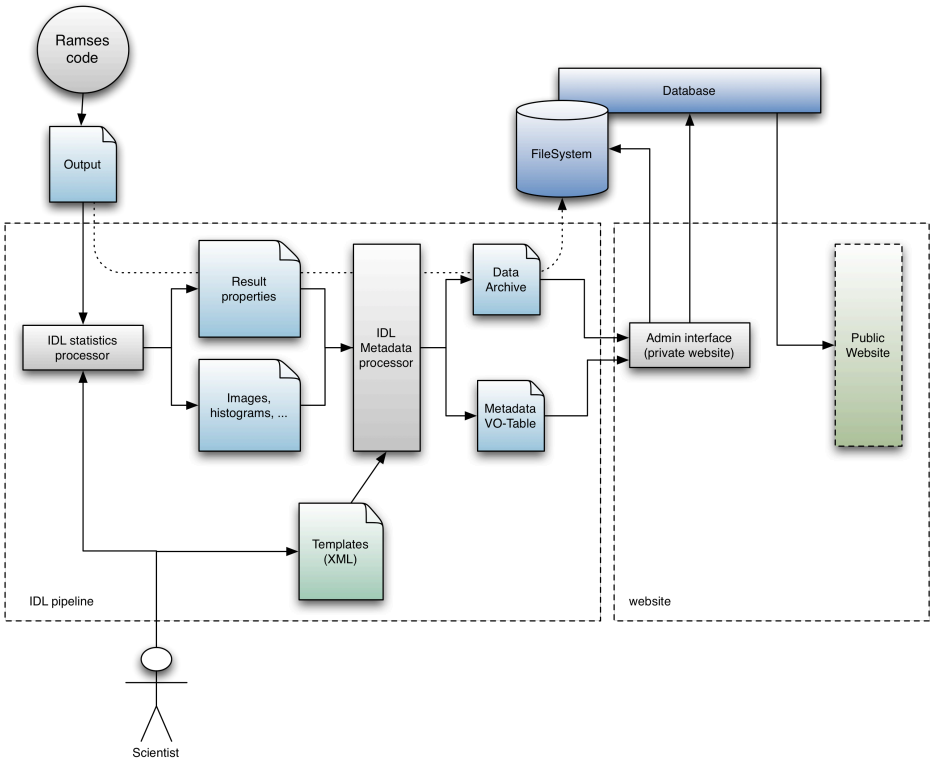
2.3.1 VO-URP environment

To implement SimDB for StarFormat, we used VO-URP. This is a Java/JPA/PostgreSQL environment deployment tool provided by the Theory Working Group members Gerard Lemson and Laurent Bourges in order to help deploy a SimDB instance or any hosting project based on any UML data model.
One just needs to provide his data model drawn with the tool MagicDraw respecting some specifications including full descriptions, and eventually provide an XSD schema of the data. Then VO-URP generates from these info a full environment with SQL scripts and JAVA/JPA classes matching the model in order to create, handle and manipulate data with the database, either in PostGreSQL or MSSQLServer.
VO-URP also provides a basic web implementation to browse through the data, import or validate data from an XML matching the model.

For StarFormat, we just had to complete the SimDM UML model within MagicDraw with a couple complementary classes in order to provide extra services functionalities for the web interface. After configuring the options for the model, compiling VO-URP provided all these tools thanks to which the full web-service could be implemented. The simple interface to browse the database (SimDB-Browser) was integrated into the administrative part of the website.

2.3.2 Ingestion pipeline

An IDL pipeline was developed in order to compute statistics, images and histograms of properties from the resulting raw data of each snapshot. This tool generates all the specified metadata and resulting files for a snapshot or post-processing and stores them within a VOTable and a tar.gz archive of extra files the theorists wants to provide along with the data. After copying these files on the server, one can import all data into the database using a simple one-click interface within the administrative part of the website. While the files are stored into a file system on the server, organized as a Project/ Simulation/Snapshot/PostProcessing/Object directory tree, the VOTable, organizes the metadata within the SimDM classes. Using VO-URP's java classes and a VOTable parser based on CDS's SAVOT Java libraries this VOTable is then read and stored as classes of SimDM into the database.



A complete JSP servlet interface was also developed thanks to VO-URP's Java classes in order to provide snapshot browsing services on the website: one can easily navigate through the different simulations and their snapshots with a synthesized summary of all recorded data. After this browsing, the user is invited to choose a selection of snapshots and query the post-processings snapshots to identify desired clumps by their properties. Users can then extract as VOTable or ASCII files all the parameters of an experiment or the properties of a clump or a set of extracted clumps from a post-processing.

Connecting the service with RAMSES post-processing command-line routines on some snapshots raw data, extraction of small subsets of resulting data is allowed to the users after identification of a desired clump in order to perform their own analysis of the results.

2.4 Comments

The only real issue that arose through SimDM's implementation was the need within StarFormat to provide histogram statistics of the molecular cloud: the simple use of the Statistics class was not sufficient to provide such feature, except by organizing the values within a property group which did not seem sufficient for a complete representation of such tables of values.
So we derived a natural set of complementary classes as StatisticsTables referring not only to one property but two. StatisticsTableValues providing values for both properties are linked tot this StatisticTable. This way, classes of statistics can be stored as well. Thanks to the VO-URP's functionalities, such change to the model is easy to handle by simply updating the UML model and running again the Java classes generation on the new model.

Apart from this specific need, the genericity and the richness of the Simulation data model allowed a good and simple description of various kinds of simulations considered within the StarFormat project. Even though we are manipulating big sets of data with a lot of characterizing properties needed to identify specific subset results of a simulation, everything could be mapped properly within a database with which a browsing and querying website could easily communicate thanks to the VO-URP project.

3 PDRDB – Micro-physics simulations

PDRDB is a SimDB service aiming at publishing in the Virtual Observatory models of interstellar clouds computed by the Meudon PDR code. The Meudon PDR code, is a 1D stationary code that computes the atomic and molecular structure of interstellar clouds. At each position of the cloud, the code computes several physical quantities as the abundance of hundreds chemical species, level excitations, gas temperature, grains charge and temperature, … It also computes properties of the whole cloud as column densities, line intensities and spectra that can be used to interpret observations.

Compared to the STARFORMAT project presented in section 2, PDRDB deals with “micro-physics” simulations.

We present here how we used SimDB to publish in the Virtual Observatory PDR models. In section 1, we present the objectives of the service, in section 2, we present the instantiation of the data model. Section 3, provides some technical information on the way we did the development of the service to facilitate its maintenance and further development. Finally, section 4 presents the advantages and the limits of the service.

3.1 Description and objectives

One of our objectives is to publish sets of models corresponding to different interstellar clouds (diffuse clouds, PDRs, extragalactic media) to facilitate the interpretation of observations in the interstellar medium. These developments were motivated by the new generation of instruments HERSCHEL and ALMA. We wish the service provides “observables” as line intensities and column densities but also the chemical and physical structure computed in the models. This is mandatory to make sure the users will have all the information to understand the physics behind the model.

The discovery of simulations is possible in two ways:

1) Search using input parameters. For example, a user can search a model of interstellar clouds with a proton density about 1000 cm^{-3} and a UV radiation field about 100 times the interstellar standard radiation field.
2) Search using outputs. For example, a user can search all simulations with the column density (or a line intensity) of species X between a minimum and a maximum value. This is useful to quickly obtain some guess to interpret observations.

Once a model has been found, through the web portal, it is possible to extract some quantities from the simulation through services in the web page. When relevant, these data can be sent from the web page to VO-Tools using SAMP. Since the number of computed physical quantities is large and heterogeneous, only the most important ones for the interpretation of observations are provided through the web portal. For the other ones, the user can download the full result of the simulation.

Some steps of the development of PDRDB with SimDB have been a challenge. The major difficulties came from the following specificities:

a) The code computes a large number of heterogeneous physical quantities:
Together the number of species, populations in levels and lines intensities represent more than several ten thousands quantities. As a consequence, we only store in the database those useful to find a model. Then, services on the web page can access to the other data stored in filesystem.
b) The code is frequently modified:
We often have to add new species in the chemistry, new physical processes in the code and we have to update frequently the atomic and molecular data used by the PDR code. Some of these modifications produce new physical quantities in the outputs that we want to publish in PDRDB. As scientists, we do not wish to modify the database each time the code is modified. The database has to update itself depending on the new ingested simulations. This has been solved structuring and documenting the outputs of the codes and developing a sophisticated ingestion pipeline using VO-URP. This is explained in section 2.4.

3.2 SimDM implementation on PDRDB

The simulation DM is an abstract DM. At some steps of the implementation, we did some choices but other ones could have been possible.

3.2.1 - Protocol part

uType: SimDB:simdb/protocol/Protocol

The protocol part defines the code used to produce the experiments. As many other microphysics codes, the Meudon PDR code can be divided in three parts:

1) Fortran source codes
2) Data files. They contain chemical reaction rates, atomic and molecular data as Einstein coefficients, collision rates, levels energies of atoms and molecules, etc. In the case of the Meudon PDR code, this represents several tens of thousands values. Some of them are well known, other ones are subject to important uncertainties.
3) Input parameters (proton density in the cloud, intensity of the UV radiation field, metallicities, flux of cosmic rays, some flags to control the algorithms or the way some physical processes are computed,)

3.2.1.1 Protocol class

uType: SimDB:simdb/protocol/Protocol

The first question is to determine if the Data have to be considered as input parameters or not.
If they were considered as input parameters, this would allow easy queries on the SimDB service to search for simulations produced with different values for atomic and molecular data. For example, it would be possible to publish sets of results with different chemical rates and to search simulations in the database with queries on some of these rates. This could be useful, but it would then be very difficult for publishers to manage a so large number of input parameters.
In the case of PDRDB, we considered that is a not a mandatory service we wish to provide in a VO context and, to simplify the implementation, we choose to not consider these data as input parameters.
As a consequence, we consider that the protocol in PDRDB is the combination of the Fortran source codes and all the atomic and molecular data and chemical reaction rates.

3.2.1.2 InputParameter

uType: SimDB:simdb/protocol/InputParameter

The InputParameter class is used to list the input parameter of the code. In our case, this corresponds to the parameters that are frequently modified at each run to explore space parameters. As mentioned above, we do not consider atomic and molecular data or chemical reaction rates as InputParameter.
InputParameters are for example the density of the cloud, the intensity of the incident UV radiation field on the cloud, the properties of the grains, the metallicities, etc.

3.2.1.3 ParameterGroup

uType: SimDB:simdb/protocol/ParameterGroup

Because the number of input parameters is important in PDRDB, several ParameterGroups are defined. They correspond for example to «metallicities» that contains the InputParameters He/H, C/H, O/H, ... or Grains properties that contains the parameters of grains (minimum size, maximum size, ...)

3.2.1.4 - Algorithms

uType: SimDB:simdb/protocol/Algorithms

The Algorithms class is used to provide some information on the main algorithms used in the code.
PDR codes take into account a large number of physical processes with very different algorithms. This class is not very useful for us because the list of algorithms used in the code is not really a way to define the capabilities of the code. More precise information is required. For example we can say that the code uses an «exact radiative transfer method» but the important information is to know if overlapping of lines is taken into account or if radiation emitted by dust can affect non locally level populations of molecules. This information is provided in the documentation of the code but not in the SimDB service.

3.2.1.5 - Physical processes

uType: SimDB:simdb/protocol/Physics

Even if not precise enough (see 2.1.3), this can be used to provide an overview of the main physical processes in the code.
Example: Radiative transfer.

3.2.2 Experiment

uType: SimDB:simdb/experiment/Experiment

An experiment corresponds to a published run of the code.

3.2.2.1 Snapshot

uType: SimDB:simdb/experiment/Snapshot

The Meudon PDR code is a stationary 1D code. We do not have snapshots with the meaning it can have in the context of cosmological simulations for instance. In the SimDB implementation, we consider that for each experiment we have a single snapshot that corresponds to the output of the code, the stationary state.

3.2.2.2 TargetObject

uType: SimDB:simdb/experiment/TargetObjectType

TargetObjects represents the astrophysical object that has been simulated. For PDRDB, most of the time this is «Interstellar Cloud». That could also be «disks» when a model has been run to simulate a protoplanetary disk.

3.2.3 Object

uType: SimDB:simdb/object/Object

This part is a heritage of the protocol and of the experiment parts. We describe first, the ObjectType and Properties for the protocol, then for the experiment.

3.2.3.1 – RepresentationObjectType and Properties for the protocol

uType: SimDB:simdb/protocol/RepresentationObjectType

At each position of the grid, the code computes several quantities for the gas and the grains. We consider two ObjectTypes : gas and grains with the attribute Mesh Cell.

The properties of the RepresentationObjectType are all the quantities computed locally. In the Meudon PDR code, we have several thousands of these properties. Only the most important ones to search models in the database are implemented.

Examples for the RepresentationObjectType, gas, with in bold face, those that are stored in the database:

- gas temperature, proton density, and ionization degree
- abundances of H, H_2, C^+, C, CO, ...
- levels populations of CO in J=0, J=1, ... of H_2 in v=0 J=0, …
- local line emissivity (erg cm^{-3} s^{-1} Ang^{-1} std^{-1}) of H_2 from v=0 J= 2 to v=0 J=0
- heating rate by photoelectric effect, by cosmic rays, ...
- density of energy per bin of wavelength
- ...

Examples for the RepresentationObjectType : grains
- size of grains
- temperature of grains
- charge of grains

Associated to an experiment, the statistics provide the mean, max, nominal values of these properties. These numbers in the database are used to provide some information on the run.

3.2.3.2 – TargetObjectType and Properties for the Experiment.

uType: SimDB:simdb/experiment/TargetObjectType

The properties of the TargetObjectType correspond to the quantities computed for the whole simulated astrophysical object. In PDRDB they correspond to integrated quantities as column densities or line

intensities. They are very important in this service because they are the quantities observers are looking for.

Example of TargetObjectType properties :
- column density of H, H2, CO, ...
- column density of H2 in level v=0, J=0, in level v=0, J=1, ...
- emergent line intensity of CO from J=2 to J=1, ...

The statistics provide the mean, max, nominal values of these properties. These numbers in the database are used to search simulations on outputs.

3.2.3.3 PropertyGroup

uType: SimDB:simdb/object/PropertyGroup

Because of the large number of properties, we group them using the PropertyGroups. So we have for example a group “abundances” that contains a list of properties, the abundances per element. In the same way, we have a group “column densities”.

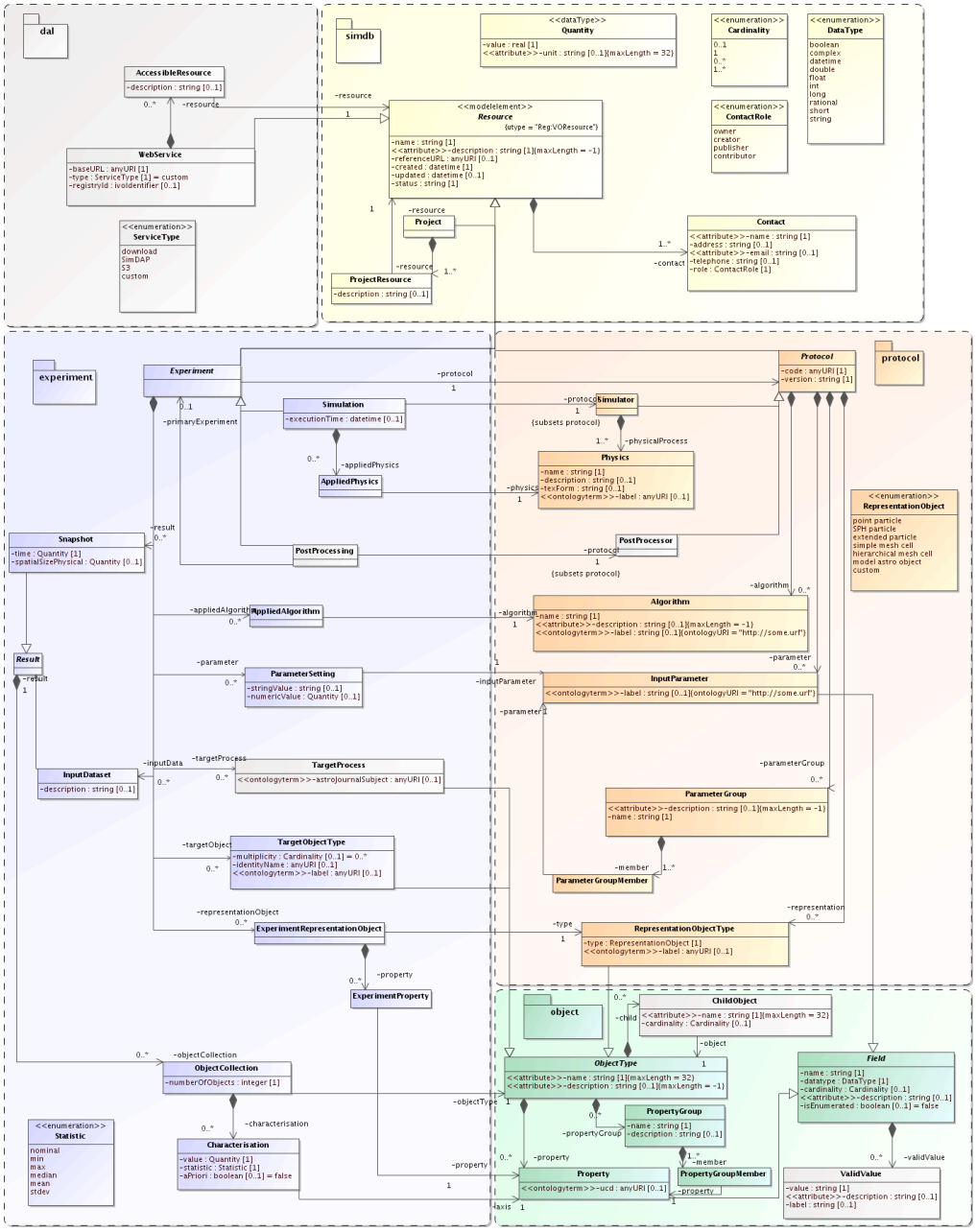
3.2.4 - Projects

uType: SimDB:simdb/Project

Projects are used to group together groups of experiments to facilitate searches in the PDRDB and its management. The way they are used is subjective.
Examples :

- Diffuse clouds : could be a project grouping a set of models with different input parameters corresponding to parameters of diffuse interstellar clouds.
- Damped Lyman Alpha Systems : could be a group of simulations with extragalactic metallicities to study the chemistry of diffuse intergalactic medium.
- Horsehead nebula : could be a project grouping all the models (from different protocols) of the horsehead nebula.

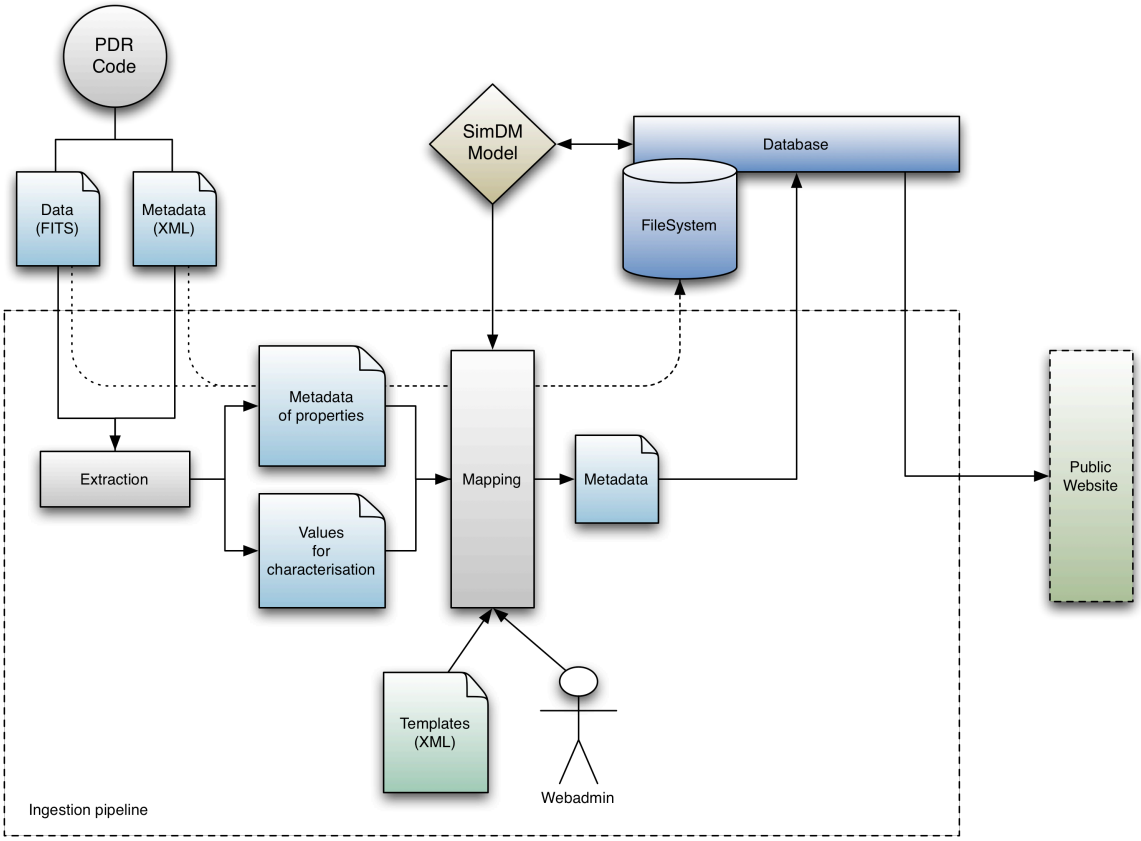
SimDB classes used for PDR's implementation :



3.3 Implementation SimDB / VO-URP

In this section we present how we implemented SimDB on PDRDB. This is not the only way this could have been done, nevertheless, this may give some ideas on how to avoid some difficulties in the development of VO services on microphysics simulations.

PDRDB is composed of :
- a database
- an ingestion pipeline
- Services



3.3.1 - Database

The PDR DataBase was built as a full implementation of the Simulation Data Model as a PostgreSQL database communicating with a Java/JPA interface thanks to the use of the VO-URP environment tool provided by the Theory Working Group members Gerard Lemson and Laurent Bourges.

3.3.2 - Ingestion of data in PDRDB

The ingestion of data in PDRDB has been one of the major difficulty from the beginning of the project: The Meudon PDR code is frequently modified to compute new physical or chemical processes. As a consequence, the SimDB service had to be flexible enough so that we avoid, each time we wish to introduce in PDRDB a new protocol or new experiments (with eventually new physical quantities in the outputs, so new properties) to have to modify manually the database.
Indeed, we want the scientist register new simulations in a very transparent way.
This difficulty should be common to many microphysics codes. We partially avoided it thanks to :

1) a modification of the outputs of the code to describe in details the list of physical quantities computed in the code.
2) the use of templates in the ingestion pipeline. These templates contain the list of the quantities that have to be extracted from the outputs of the code and imported in the database. They are in XML format and can be easily modified by any scientist.

3.3.2.1 Metadata describing the outputs of the PDR code

To develop PDRDB, we had to modify the output format of the PDR code. Previously the code produced a binary file containing all the computed quantities. This binary file could be difficult to read on different architectures and the physical quantities were not properly described in it.

We modified the PDR code so that it produces 2 output files :

1. a FITS file containing the data. FITS has been chosen because :
    i. it is a format easy to read on any architecture
    ii. data are structured in tables and so it is simple to extract specific quantities or to write new ones in a structured form.

    Note that FITS may not be the best choice because of its limitations. HDF5 or some other formats could have been more appropriate.
2. a XML file describing the content of the FITS file. This XML file follows the syntax of VO-TABLE because many routines have been developed in the VO to deal with them (savot for example).

An ID that is present in the FITS file as keyword and in the XML file identifies each quantity. Associated to this ID, the XML file contains a human readable name of the quantity, a full description, units, UCDs, …

These two files contain all the quantities computed by the code, the input parameters of the model, and some of the atomic and molecular data used to produce the simulation.

The XML file also provides the organization of properties as groups. Each group has also an ID. For example, there are hundreds properties “ABUNDANCE_X” where X is the name of the chemical species and a group ABUNDANCE.

3.3.2.2 – Templates and ingestion pipeline

To document the simulations within the database all results from the code are not necessary, so we first made choices on which parameters and properties to give access to and wrote a protocol and a experiment XML templates giving a short description of all basic groups of elements to include within the database for a version of the code and a run.
Thanks to the JSP servlet interface developed with VO-URP's JPA classes to communicate with SimDB, we created an easy administrative interface which allows adding of new code versions and new models in a few clicks : after adding textual descriptions and placed the XML+FITS output within a specific directory, the ingestion pipeline matches the model fields along the templates definition, reads the values and stores them within the correct classes of the data model.
Thanks to this interface, whenever we want to introduce new quantities in PDRDB, one only needs to add the corresponding ID in the template file.

3.3.3 Services

In order to help users browsing data, we provide them some tools for visualization and interoperability. Concerning the spectra :

3.3.3.1 – Visualization

The PDRDB website allows the user to display, compare and save spectra available for a given experiment. This functionality is a java applet implemented thanks to the JFreechart library. Getting and plotting data is a 2-step process.
First of all we query the server to know which types of spectra are available for a given PDR experiment (incident or emergent with different angles for example). The response is an xml document. Then the applet parses this document to create its own GUI according to the type of spectra available.

The user can now query the service to get a spectrum. It is returned by the server as a simple csv file containing 2 columns (wavelength/intensity). The plotting is managed by the JFreechart API (we use it to draw histograms on the Starformat database too).

The data can be saved as ASCII or VOTable files.

3.3.3.2 - Interoperability

All the displayed spectra can be sent to another VO applications through a SAMP hub. This functionality has been implemented thanks to the WebSampConnector library developed by VO-Paris Data Centre. It consists in an java applet that can connect to a SAMP hub, send and receive messages, and some JavaScript functions that create the interaction between the web page and the applet.

To send data from the visualizer to the SAMP applet, we have to establish a communication between the 2 applets. This is achieved thanks to the JSObject (netscape.javascript.JSObject).
This object can call JavaScript functions from the web page containing the applet :

```
//array containing the parameters of the JavaScript function
String[] params = {"v", id, id, votableUrl};
//call the function directly by its name
jsInterface.call("sendSampMsg", params);
```

All the selected spectra in the interface are then sent to all the VO applications connected to the hub.

Appendix A: SimDB Classes Instantiation

The table below summarises some key elements defined as instances of the different classes of SimDB within the two different implementations.

	PDR	Starformat
Project	Diffuse Clouds PDRs Extragalactic Clouds	Molecular Clouds Dense cores Turbulence
Protocol	PDR 1.4 – Chemistry10 PDR 1.4 – Drcnosd	Ramses-MHD 2 Ramses-MHD 3
RepresentationObject	AMR cells (gas, grains)	AMR cells
TargetObject	Interstellar cloud	Molecular cloud
Snapshot	Single result (stationary)	Snapshot (time iteration)
Property (Represent.)	Abundance H, H2, … Gas temperature …	n/a
Property (TargetObj.)	Column density H, H2, … Line intensities	Statistics on: Temperature, mass, density, …
PropertyGroup	Abundances Column densities	n/a
PostProcessing	n/a	Clump extractions
TargetObject	n/a	Clump
Property (TargetObj.)	n/a	Temperature Mass Density

References

To be done once SimDM will have its url.