International

Virtual

Observatory

Alliance

Simulation Data Model

Version 1.00-20100520

IVOA DM WG and TIG Proposed Recommendation 2011 May 20

This version:
1.00-20110520
Latest version:
1.00-20110428,
http://volute.googlecode.com//svn/trunk/projects/theory/snapdm/specification/WD-SimulationDataModel-v.1.00-20110428.doc

Previous version(s):
See revision page on GoogleCode:
http://code.google.com/p/volute/source/browse/trunk/projects/theory/snapdm/specification/

Working Group:
http://www.ivoa.net/cgi-bin/twiki/bin/view/IVOA/IvoaDataModel
http://www.ivoa.net/cgi-bin/twiki/bin/view/IVOA/IvoaTheory
Editors:
Gerard Lemson, Hervé Wozniak

Authors:
Gerard Lemson, Laurent Bourgès, Miguel Cerviño, Claudio Gheller, Norman Gray, Franck LePetit, Mireille Louys, Benjamin Ooghe, Rick Wagner, Hervé Wozniak

---

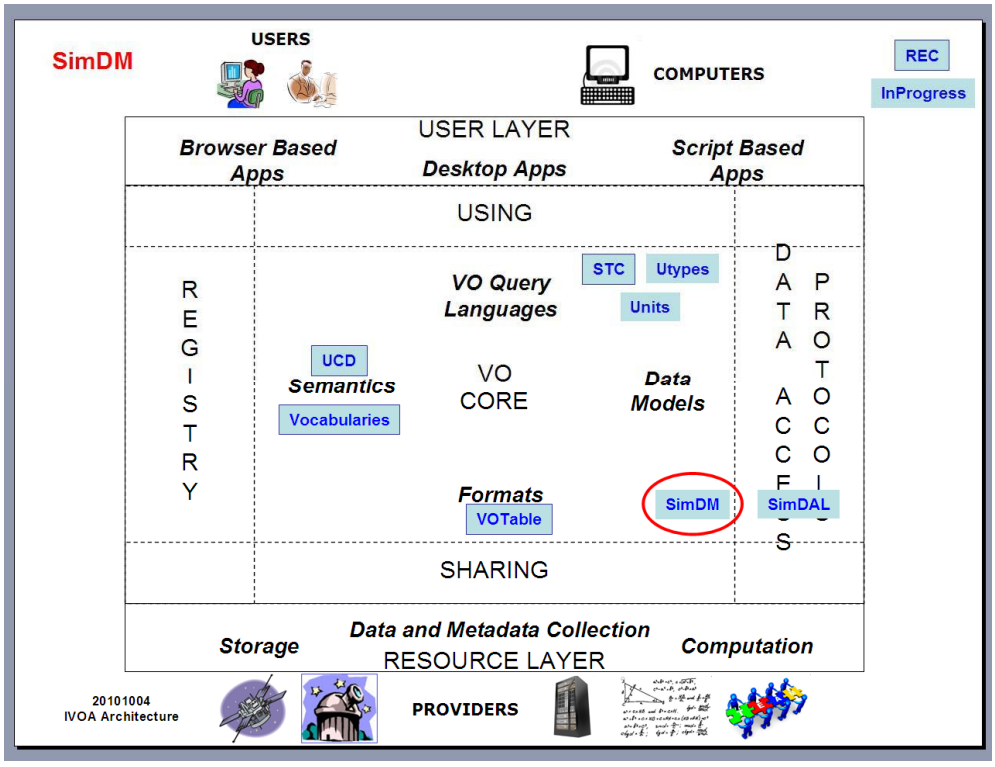
Abstract

In this document and the accompanying appendix we propose a data model (Simulation Data Model) describing numerical computer simulations of astrophysical systems. This data model is designed to support various IVOA protocols under construction for discovering and accessing the data products resulting from such simulations and their possible post-processing. One of these is the "Simulation Database (SimDB)" protocol (hence the former name SimDB/DM for SimDM), which describes a particular web service that gives access to a database containing metadata describing simulations. A separate document, "*Simulation Database*: *Serializations and Services*" (referred to as *SimDB Services* in the rest of this note) will cover the physical representations of the model as used in the SimDB protocol, and how they will be used in the SimDB interface.

The SimDM has been developed in the IVOA Theory Interest Group with assistance of representatives of relevant working groups, in particular DM and Semantics.

Link to IVOA Architecture

The figure below shows where SimDM fits within the IVOA architecture:



Status of This Document

This is an IVOA Proposed Recommendation made available for public review. It is appropriate to reference this document only as a recommended standard that is under review and which may be changed before it is accepted as a full recommendation.
The first release of this document was 2011 April 28.

A list of current IVOA Recommendations and other technical documents can be found at http://www.ivoa.net/Documents/.

Acknowledgements

We thank various persons for useful discussions in the course of this work. First the participants of the Feb 2006 theory workshop in Cambridge, UK, where this work was started. Second the participants of the April 2007 SNAP workshop in Garching, Germany, where the design started taking shape. The work has also been influenced by the participants of the Technical Coordination Group of the EuroVO-DCA project and participants of the theory workshop organised in the context of that project in Garching, 2008. Then we want to thank particularly the following persons for useful discussions and feedback: Jeremy Blaizot, Miguel Cerviño, Klaus Dolag, Pierro Madau, Adi Nusser, Ray Plante, Volker Springel, and Alex Szalay. We finally want to thank participants to the theory sessions in all the interoperability meetings since Victoria 2006, where parts of this work were discussed.

Conformance related definitions

The words "MUST", "SHALL", "SHOULD", "MAY", "RECOMMENDED", and "OPTIONAL" (in upper or lower case) used in this document are to be interpreted as described in IETF standard, RFC 2119 [0].

The **Virtual Observatory (VO)** is a general term for a collection of federated resources that can be used to conduct astronomical research, education, and outreach. The **International Virtual Observatory Alliance (IVOA)** is a global collaboration of separately funded projects to develop standards and infrastructure that enable VO applications. The International Virtual Observatory (IVO) application is an application that takes advantage of IVOA standards and infrastructure to provide some VO service.

Contents

1 Executive Summary

In this document we make a proposal for an IVOA standard data model for describing simulations[1]. Indeed, apart from limited support for publishing model spectra in SSAP, there is as yet no IVOA standard dealing with the publication of simulations and their results. The primary goal of our proposal is to support discovery of simulations by describing those aspects of them that scientists might wish to query on, i.e. it is a model for *meta*-data describing simulations. This note does *not* propose a protocol for using this model. Two distinct IVOA protocols are in the make and both are supposed to use the model, either in its original form or in a form derived from the model proposed here, but more suited to the particular protocol.

The direct motivation of the model comes from the Simulation Database (SimDB) and the former Simulation Data Access Protocol (SimDAP) efforts. Work on these standards started under the header *Simple Numerical Access Protocol* (SNAP) in the Theory Interest Group (TIG) since the Victoria interoperability meeting, 2006. It was agreed in the Trieste interop 2008 to split SNAP in two separate tracks, SimDB and SimDAP. More recently it was deemed useful to further split the work on SimDB in two tracks, one focusing on the data model alone, the second on its serialisation and usage in the SimDB protocol. This document deals with the data model, referred to as SimDM.

Work on the SimDM (and SimDB) specification has been organised via a GoogleCode SVN repository in the *volute* project originally created by Norman Gray for the Semantics Working group. The history of the SimDB project can be obtained from https://volute.googlecode.com/svn/trunk/projects/theory/snapdm. A large part of that repository deals with technical issues revolving around a code generator we designed to derive relevant resources from the basic data model. Most of that development has been moved to a separate GoogleCode project, VO-URP. The resources dealing with the SimDM developments have been gathered in the subdirectory in https://volute.googlecode.com/svn/trunk/projects/theory/snapdm/specification/..

In Section 2 we recall the scientific motivation at the origin of creating SimDM and the 4-years history of the developments. Section 3 described our methodology whereas the model itself is detailed in Section 4. A few specific issues of serialization are addressed in Section 5. The development of SimDM is linked to other IVOA efforts that deserve to be mentioned. Section 6 deals on that point. We present some conclusions and describe the next steps for the recommendation process in the last Section.

[1] We will use the term *simulations* for the running of a simulation code as well as for their results. And we will often include post-processing codes and their results as well.

2 History

Numerical computer simulations form an increasingly important component of astrophysical research. Such simulations are used to model astrophysical processes whose complexity precludes an analytical treatment. The subject of these simulations includes every possible astrophysical phenomenon, from the structure of stellar atmospheres, the formation of solar systems, the structure of galaxies and the description of their constituents, to the formation of the largest structures in the universe.

The simulations often result in predictions that can be compared to observations, but in general are much richer, including “observables” that can only be derived by indirect means from observations. These results can be very large, rivalling and often exceeding in size the largest observational catalogues. But they can also be relatively small, consisting of individual spectra of say a white dwarf, though often in collections resulting from parameter studies.

The design and execution of these simulations has become a specialised field of astrophysics, and is these days often performed in large collaborations. And while it is still true that their results are studied by these groups only, more and more of these theoretical data are being published online (see for instance the Appendix B of [26]).

Apart from limited support for publishing theoretical spectra in SSAP, there is as yet no IVOA standard dealing with the publication of simulations and their results. In earlier documents we have described the issues for defining such standards compared to the arguably simpler case of observational data sets (see for example [21] and [26]).

The proposal for a standard way of publishing simulations was formulated during a workshop in Cambridge, February 2006. The original idea was to create an analogue of the simple image access protocol (SIAP, [18]) for N-Body simulations: SNAP, the *Simple Numerical Access Protocol*. During the following interoperability meeting in Victoria, May 2006, the scope was expanded to include other types of simulation algorithms, and rephrased to something like “simulations that reproduce 3+1dimensional space time”. It was felt furthermore that not only simulations themselves should be included, but also certain types of post-processing such as cluster finders, as long as their results are still aimed at producing a description of 3D space at one or more points in time. Over time requests have come in to generalise this scope even more, basically to enable any type of astrophysical simulation to be handled.
An important change that was decided in Victoria 2006 was that instead of the SIA protocol, the newer simple spectral access protocol (SSAP, [19]) should be followed as an example. This protocol’s main difference with respect to SIAP was

the explicit data model that was created for spectra and was used as motivation for the queryData metadata and the getData data format. Hence SNAP from the beginning had a double focus on a data model plus related query protocol on the one hand, and a data access and delivery specification on the other hand.
Shortly before the Trieste interop in the spring of 2008 it was decided to split SNAP up along these lines in two separate specifications. A specification for a *Simulation Database* (SimDB) which would support searching for interesting simulations and the services providing access to them, and a *Simulation Data Access Protocol* (SimDAP) providing a specification for accessing simulation results.

SimDB on its own is still a rather complex specification. It has overlap with the efforts and results of many working groups, Data Model (DM), Registry, Data Access Layer (DAL), Semantics as well as being an integral part of the Theory Interest Group (TIG). This issue has been discussed in the Baltimore and Strasbourg interops, as it causes a potential problem for the standardisation process: an interest group cannot promote a document to a standard, but which a working group (WG) could do so. It was decided in Baltimore to postpone that decision by creating a focus group led by the TIG and with participation form the various WGs.

The current document is the result of a split in original Note that was written for SimDB. Such a split was proposed to simplify the standardisation process and after some refactoring was performed mid-2009. This current document is the first of these and deals exclusively with the data model (SimDM) and consequently has a natural place in the DM WG. The second document deals with the use of the data model for defining the model for a relational database and its related TAP query implementation as well as a service interface for uploading simulation descriptions to this database. It is not yet clear whether it can find a place in a single WG.

A recent effort has been the proposal for a simpler access standard for small scale simulation, the Simple Self-describing Service protocol (S3, [15]). This was a result of an investigation started in the Cambridge 2007 interoperability meeting whether “micro-physics” simulations as they are sometimes called require special attention. For some time this was covered by SSA, at least as far as theory spectra were concerned. S3 is actually a direct reworking of an older Theoretical Spectral Access Protocol [20].

There were questions in the TIG whether S3 might be incorporated in SimDB and/or SimDAP. In the interoperability meeting in Victoria 2010 the decision was made that indeed this should be possible. The SimDM was shown to be able to incorporate the metadata for S3-like services, and indeed proposes extensions of that. It was decided that the S3 protocol should be merged with/incorporated into the SimDAP standard, which from then on will be known by the name Simulation Data Access Layer (SimDAL). The appendix document addresses this question

from a formal point of view, namely by defining *how* S3-like services can be described by the data model.

3 SimDM: application, approach and outline

3.1 Application(s) of the model

The data model proposed here was never meant to be created in a vacuum, but was always intended to be used in some IVOA standard service. What precisely this application was has not always been clear for the SimDM, the goal of which has gone through some changes in the course of the project. It started off as a SIAP-like service protocol for N-body simulations, SNAP [2]. Then mesh simulations were included, leading to our definition that SNAP should support the discovery and possibly partial retrieval of simulations involving "evolving objects in 3D space"[3]. Over time the protocol separated into the Simulation Database, a more TAP service for querying for simulations, and the SimDAP protocol for retrieving actual data products[4]. Recently (and finally) also simulations of a different type, "micro-physic simulations"[5], or "models"[6] have been included, and SimDAP has merged with S3 to form SimDAL: a family of access protocols for theory data[7].

Consequently the aim and possible applications of the data model have changed a little over time. The discussions in the Victoria 2010 interop have finally led to a convergence of these ideas and to the following agreement:

1. The SimDM MUST support the SimDB protocol and the SimDAL service protocols.
2. To do so it MUST allow scientists to describe their simulations in sufficient detail for others to decide whether a given simulation is of interest, and to query for these simulations.
3. It SHOULD then offer directions to services for further drilling down or downloading simulations or parts of these.
4. The precise representation of the model in the individual service protocols (SimDB, SimDAL) is not prescribed.
5. If individual protocols will deviate in the details from the SimDM, this will be for application specific reasons.
6. The SimDM will provide the vocabulary for the concepts used in these possible alternative representations.

In the preferred case, these other representations can be considered as different *views* on the core SimDM. This should be interpreted similar to the way relational

[2] http://www.ivoa.net/cgi-bin/twiki/bin/view/IVOA/CambridgeTheoryWorkshopFeb06
[3] http://www.ivoa.net/internal/IVOA/InterOpMay2006Theory/closingplenary.ppt
[4] http://www.ivoa.net/cgi-bin/twiki/bin/view/IVOA/InterOpMay2008Theory
[5] Coined in Cambridge interop, 2007.
[6] In Victoria 2010, we decided to label all as simulations. See http://www.ivoa.net/internal/IVOA/InterOpMay2010Theory/IVOA2010_ModelvsSim.pdf
[7] http://www.ivoa.net/cgi-bin/twiki/bin/view/IVOA/InterOpMay2010Theory

database views provide a mechanism by which slightly different representations from the often more normalised data model can be provided.

3.2 Modelling approach

With this application area in mind we have followed a somewhat formal approach to our data modelling effort, suggested by e.g. [28]. In this approach the construction of a data model is divided in three stages. The first is the *analysis* stage, in which one investigates the "domain of the application" one tries to model. This stage produces a *conceptual*, or *domain model* [33] which is relatively abstract and high level. In its design one does *not* aim to create a model directly suited to an application. The emphasis is on identifying important concepts and their relationship in the "real world". One also refrains from giving a fully detailed definition of all attributes and other features. Important in this stage is interaction with *domain experts*, in our case scientists.

The aim of the second stage is the creation of a *logical model* [34] of the application domain. This should be a detailed model supporting the application. It should contain all information required to support the application. It should however still be implementation neutral, and concentrate on describing the precise concepts and semantic relationships between them. In general it will use part of the concepts from the domain model, but works them out in more, all, detail.

In the third stage one derives from this logical model one or more *physical models* [35]. These are representations of the logical model in a form that can be used directly by the various computational components building the system. Examples of this are XML schemas defining valid XML documents, or relational schemas for the design of a database storing instances of the model.

The analysis phase is described in the next few subsections. The logical model is defined in full detail in UML using the MagicDraw Community Edition modelling tool. MagicDraw stores the model in an XML file following the XMI serialisation [30] we have created a fully cross-linked HTML file documenting this model in all detail. These 2 files are part of this specification and can (for now[8]) be found on the GoogleCode volute site.

3.3 Phase 1: analysis

The analysis phase investigates the "world the application lives in", its "universe of discourse" [28] and describes it in a domain model. To get constraints on this universe and its contents we follow [36] in trying to gather some 20 science questions that the application should be able to answer. The application is here a system consisting of the data model together with the protocol and implementations. The model will be designed in such a way that it can contain the required information. The protocol and implementations must support efficient

[8] Once the specification becomes a working draft the stable versions of these files should be placed under the IVOA wiki site.

querying for this information. [36] used this approach in the design of the SDSS database.
To create such a list of questions we have contacted scientists with the question that if they were presented with a database of simulation metadata, what questions they would want to ask of it to find interesting simulations. The following list summarises their answer:

- What system/object is being simulated?
- What physical processes are included?
- How is the system being represented in the simulation (particles (Langrangian), (adaptive) mesh (Eulerian)), both, other?
- How are the physical processes implemented?
- What numerical approximations were used (e.g. resolution, softening parameter)?
- What observables are available for the system/object, possibly as function of time[9]? As it is a spatial system, at least *simulation box*size, center-of-mass position.
- What observables are available for the constituents, i.e. what is the schema of the objects from which the simulation built e.g. particles in N-body simulation, grid cells in an adaptive mesh simulation or particle groups in a cluster finder?
- Per snapshot, per simulation object type, per variable:
    - Characterise the *possible* values
    - Characterise the result
- Are post-processing results available?
- Are services/applications available for accessing the results?
- Which code ran the simulation?
    - Which *version* of the code?
    - Is software available?
- Who ran the simulations*?*
- What were values of input parameters?
- How were initial conditions created?
- How the results are parametrized?
- Can I access grids of models? Can I access individual results?
- Which are the inputs ingredients (usually, which data collections are used?)
- How I can run a simulation? Can I do it on-the-fly?
- Can include my simulations in the VO in an easy way? What I should do?
- Can i compare different simulations? Can I compare the simulation with my data?
- Which simulations provide diagnostic tools? (i.e. distance/extinction/quasi-scale free quantities)
- Can I combine the results of different simulations in a single file adapted for my needs (e.g. own code)?

[9] Re: Rick Wagner's example of certain properties only being calculated after a certain stage in the simulation is reached.

3.4 Phase 2: domain model

The result of the analysis phase is a model in its own right, albeit rather sparse and schematic. For this purpose we have built on previous work by adapting the so called *Domain model for Astronomy* proposed in [12]. This model forms the basic structure of the domain model for SimDM, illustrated in Figure 1.

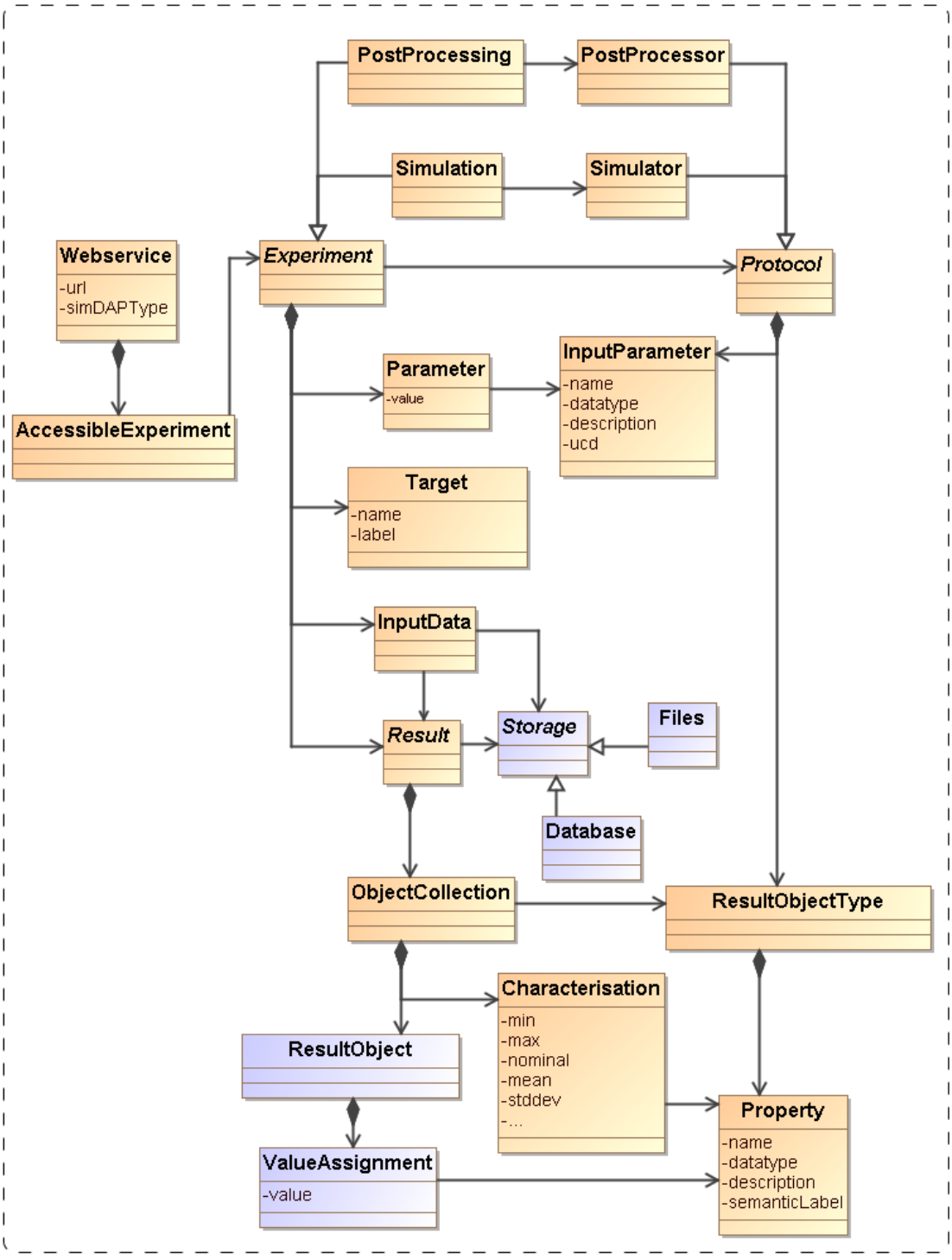


Figure 1 Schematic domain model encapsulating the main design constructs in SimDM. Elements coloured orange are represented directly in SimDM, possibly with a different name. Purple elements are not part of that model, but are used to explain and motivate other features that do appear there.

Figure 1 is used in a narrative motivating the final structure of the full SimDM. We start by assuming the existence of one or more **Files** that a publisher thinks may be of interest to the community because they contain astronomical data. Instead of in files the data might also reside in a **Database**, and to be generic we introduce a **Storage** base class that abstracts the actual physical location of the data.

Registering that files exist somewhere is not of great interest without providing information about the *contents* of the files. The philosophy that we follow is that the files are of potential interest because they contain the **Results**[10] of an (astronomical) **Experiment**, and accordingly their contents must be explained by describing the experiment that gave rise to it. Only in this way can one make scientific use of the files or other storage resources.

The abstract **Experiment** is made concrete by adding some examples of experiment types that are important for the current model dealing with **Simulations** and simulation **PostProcessing**.

In our model, **Experiment** represents the actual *running* of an experiment; to describe the *design* of the experiment (the so-called *experimental protocol*) we introduce the concept of *(experimental)* **Protocol**[11]. This separation between design of experiment and the execution is a *normalisation* that reduces redundancy in the model. See the accompanying appendix for a discussion of this concept. We mirror the concrete subclasses of **Experiment** by adding concrete subclasses to *(experimental)* **Protocol** such as **Simulator**, which represents simulation codes according to which **Simulations** are run, and **PostProcessor** corresponding to **PostProcessing** runs.

The *(experimental)* **Protocol** class contains **InputParameters**. An **Experiment** using a particular *(experimental)* **Protocol** only needs to indicate the *values* for these parameters. In this way a single instance of the *(experimental)* **Protocol** can be reused by many **Experiments** performed according to it.

The *(experimental)* **Protocol** also defines the possible structure of the results of the experiments. In our model **Results** contain **ResultObjects.** These objects have a given type, represented by the **ResultObjectType** contained by (*experimental)* **Protocol**. The **ResultObjectType** defines the **Properties** that these objects have.

For example the results of N-body simulations may contain particles having properties position, velocity, mass and possibly others. Adaptive Mesh Refinement (AMR) simulations produce results that are collections of mesh cells of various sizes, positions and contents. Similarly post-processing codes such as

[10] We do not assume that in reality the relation between the conceptual Result and the concrete Storage elements can be modelled by a single reference. Especially for the largely non-standardised world of simulations a single result can be distributed over many files, but it is also possible for one file to contain multiple results. In the current SimDB model we do not attempt to model such relations explicitly. We delegate the responsibility for accessing the physical results to (web) services and this issue is more explicitly addressed by the SimDAP protocol.

[11] Further on, the word *protocol* will be preceded by the adjective *experimental* (in parenthesis and italicized) to keep clear the distinction with any other IVOA protocol.

halo finders produce “halos” and “semi-analytical” galaxy formation codes produce galaxies.
In general a single result can contain objects of different types. For example a Smooth Particle Hydrodynamics (SPH) simulation may contain dark matter particles, star particles and gas particles. And in general the codes allow one to configure which of these exactly are chosen in a given experiment.
One aspect of the experiment that is not determined by the experimental protocol is *why* the experiment was performed. In the model we introduce the **Target** concept for this, which represents real world objects or processes that are being simulated. For example, with the same N-body simulator one may simulate a galaxy merger or the evolution of large scale structure of the universe.
As discussed above, the actual way in which results are stored in files or databases is hard, if not impossible to model. Instead we assume that **Webservices** of various kinds may be used to access the results of simulations and other SimDB products.
Some of these will be standardised in the SimDAL specification, but custom services may also be introduced. The model allows one to describe the experiments and their results, which should allow users to discover results of interest, after which the web services can be called for actually accessing these.

4 Logical model overview

The actual DM that we propose here is a logical model in the sense of [34] based on the domain model described in 3.4. It is still implementation neutral, but it is fully detailed and represented in UML. It has a human readable HTML representation which contains the detailed description of all elements [6]. That document should be consulted for the details of the model.
Here we introduce the main concepts and motivate the main design decisions. For a detailed description one must consult the accompanying HTML documentation in [6]. Where possible we try to give a link into that online document the first time we describe a concept from the model. This will be bold-faced and consist of a root URL to the location of the HTML document, followed by a #<UTYPE> that identifies the description of the actual concept in the HTML document. This we feel is very much in the spirit of the use cases of UTYPEs as described for example in [17].
In general we represent class names prefixed with “SimDM/” rather than giving the full UTYPE. As class names happen to be unique in the model, not just in their packages, this should present no confusion. We use the prefix to make clear when we refer to the actual class, as opposed to a more general concept of the same name.

4.1 Packages

UML Packages are subsets of classes and data types that are deemed to belong together. Whilst not essential to the model, we have used them to provide some level of modularity. Their main role is played in the XML schemas derived from

the model. Each package has its own type-schema (see 5.2) which provides a somewhat finer level of reuse.
The diagram in Figure 2 shows the packages we use and their dependencies. The **resource** package contains the **protocol**, **experiment** and **dal** packages which define specialisations of classes defined in resource. It depends on the **object** package, which contains classes defining objects hierarchies. All of them also depend on the **meta** package, which contains some very basic types such as **Quantity** or **DataType**. This hierarchy is reflected in the UTYPEs, see section 5.1. The colours assigned to the packages correspond to the colours of classes in the diagrams in later sections. The subdivision in the one parent and four child packages follows the resource class hierarchy described next.

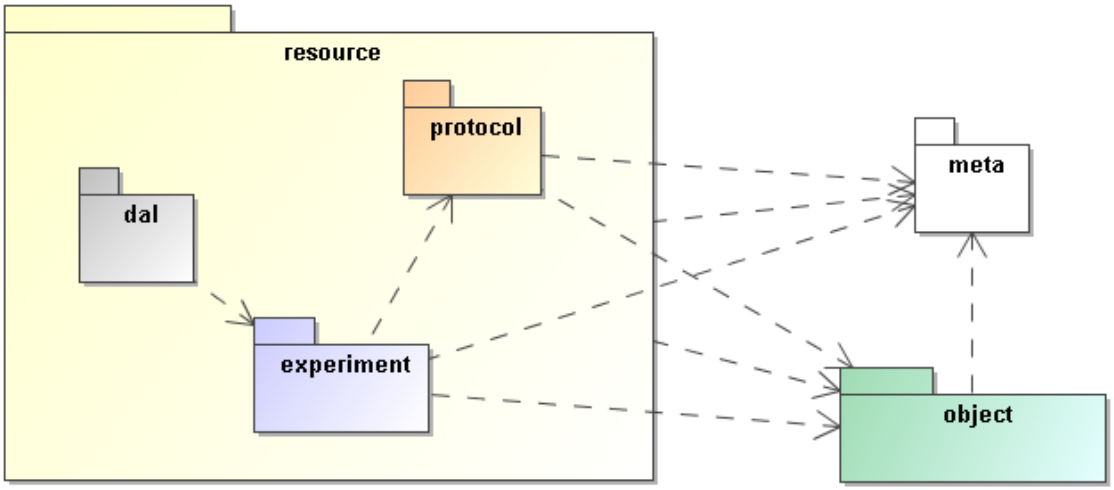


Figure 2 The packages of the SimDM and their relationships. The resource package is the parent of three "child" packages. These are related to each other through directed dependency links indicated by the dashed arrows. Apart from the dal package the others depend on the meta and object packages.

4.2 Resource

The SimDM aims to describe simulations and related concepts. The current model does so with of the order of 40 separate object types, or classes. Most of these classes themselves represent parts of other classes. They group together properties or relationships used in the definition of their "parent". The composition relation is used to represent these kinds of parent-child relationships [2]
But among the classes in the model there are some that are not used like this. These classes represent concepts that can stand on their own, are not use to describe part of a larger concept. These we will call "root entity classes". In the model they can be identified by the fact that neither they, nor any of their sub or base classes are part of another class, a child in a parent-child relation.
These are the classes that represent the model's core concepts and their identification is a first important choice in the modelling effort. In the current model there actually two separate collections of classes that are root entities.

There is the Party class, which represents an individual or organisation, and is not so important for the moment. We will focus here on the root entity classes in the Resource hierarchy, illustrated in Figure 3.

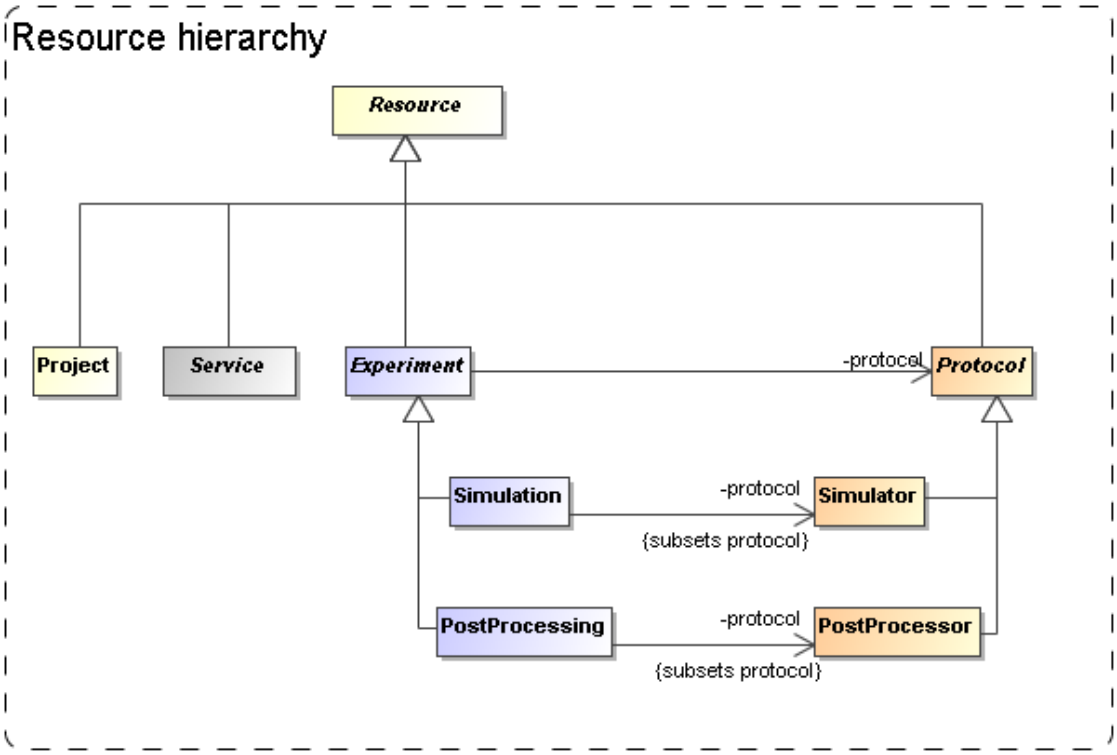


Figure 3 Root entity classes for SimDM.

From the bottom up, we have concrete classes **SimDM/Simulation** and **SimDM/Simulator**, **SimDM/PostProcessing** and **SimDM/PostProcessor**, **SimDM/Project** and **SimDM/Service**.
Simulation and PostProcessing are both subclasses of the abstract ***SimDM/Experiment*** and Simulator and PostProcessor are both ***SimDM/Protocols***. All the entities are ultimately subclasses of ***SimDM/Resource***.
Our choice for the root entities follows the domain model in concentrating on the scientific experiments as a whole. The Experiment class contains classes representing the actual results (**SimDM/OutputDataSet**) that people may wish to access, but those are *not* the core concepts in our model. This is in contrast for example to the spectrum data model [11]. Part of our motivation is that an experiment can exist without having (yet) produced any results, but to have results (as defined here) one always needs an experiment. This is a clear example of a parent-child dependency, where the child's life-cycle depends on that of the parent. The standard way to model such relationships is using a composition relation and that is how we have modelled it. More about the way we model results below in 4.7.

The separation between *(experimental) Protocol* and *Experiment* is another important feature that we directly take over from the domain model. This design was already motivated in Section 3.4 and is related to the Measurement-Protocol pattern in [27]. That pattern says that when one does a *measurement* (of some property) it is important to remember the *protocol* by which the measurement was made ([27], p65). In [12] this was extended to experiments, which in general consist of large numbers of "measurements", all done in similar ways. Whereas the term measurement seems to be more applicable to observations, it is simple to generalise the concept a bit and apply it to the *calculation* of properties during a simulation. Actually this is similar to the CalculatedMeasurement in [27],

An important reason to keep this separation between Experiment and *(experimental)* Protocol also in our logical model is to avoid having to redefine the parameters and other aspects of a simulation code each time a simulation is run.

The Service did not appear in the original domain model in [12], but we introduced it in the model in 3.4 under the name WebService. In our model it represents a way to provide access to results of experiments. We could have tried modelling the way results are stored in files etc, but deemed it too complex to do so. This is in contrast for example to the spectrum data model, where we can model the data directly and even can predefine the representation of the data. There an access reference to the data files can be given to download a result. For simulations this is in general not possible. In many cases simulation codes have their particular proprietary formats, often storing single results over multiple files. Hence we merely allow users to describe services by which one can access results, but leave the details to the SimDAL service specifications.

The Project class represents an aggregation of Experiments and *(experimental)* Protocols that combine to define a scientific project. This class is introduced to allow for example publishers to group simulations and post-processing runs that were produced with a common goal. It was inspired by a discussion on whether some of the SimDM/Resources could be registered as Registry Resources as well[12]. Many of the simulations registered in a SimDB will not qualify for the same reasons that individual images do not qualify to be registered. Resources in an IVOA compatible registry are relatively coarse grained; correspond to archives full of images published through a SIAP service for example. A Project can be used to define such collections also in SimDM. And indeed one may wish to register such collections separately in a registry.

The root of the hierarchy of entities is formed by the Resource class. This class is introduced as a convenience to hold on to information common to all its sub classes. Its name is obviously inspired by the Registry's Resource [13] and it also holds on to curation information. It "is not a" Registry Resource though in the strict OO modelling sense. For example it does not inherit all features of that class. But this is mainly because, as mentioned above, most SimDM Resources will not qualify as Registry Resources.

[12] Thanks to Ray Plante for his contributions to this discussion.

4.3 Physics, models and algorithms

An important characteristic of simulation codes is what physical systems and processes are, or can be, modelled and how these are represented in the program. The Simulator class represents computer codes that create numerical models of the world. Simulators do so by representing physical processes using numerical algorithms that act on model representations of real world objects. In our model, see Figure 4, physical processes are represented by the **SimDM/Physics** class. It is contained in the SimDM/Simulator class, not in the general SimDM/Protocol. In effect a SimDM/Simulator is distinguished from other SimDM/Protocol-s in that it models and implements physical processes.
Physical processes are implemented using particular **SimDM/Algorithms**. Other *(experimental)* protocols also have Algorithms, but these are assumed to process existing results, and do not model physical processes. Examples of these are particular algorithms for extracting clusters from results of N-body simulations.

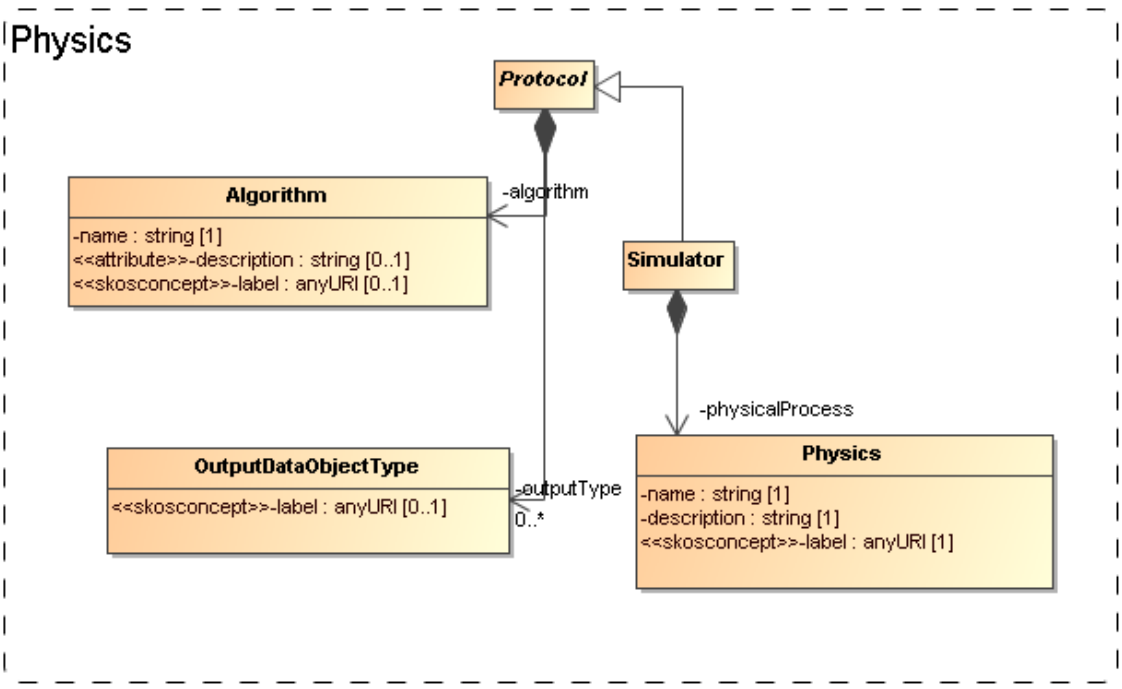


Figure 4 Modelling the representation of physical processes and objects.

Finally, *(experimental)* protocols need objects to represent the structure of the physical systems they model. For example, N-body simulations need particles that represent mass moving around. The model uses the **OutputDataObjectType** for this. This class defines the smallest components that are being used by the algorithms and are generally stored in the results of the experiments. Note that much of the structure of the OutputDataObjectType is defined on a base class, **ObjectType** that will be discussed in 4.6 below.

4.4 Parameters: definition and values

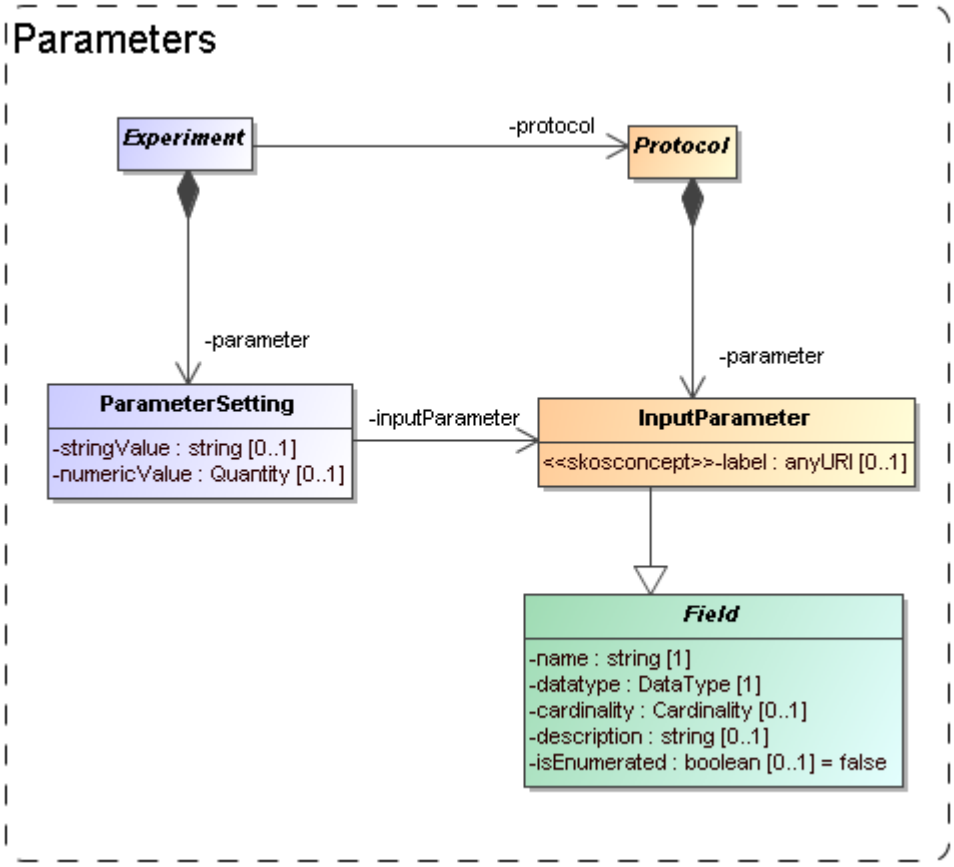


Figure 5 Modelling the parameters: definition under *(experimental)* Protocol, values under Experiment.

Software codes generally require some level of configuration before they are executed. In many cases this translates into a collection of parameters that must be given values. The parameters are defined by the code and we model this by an **InputParameter** class that is contained by *(experimental)* Protocol. Assigning values to these parameters however is the responsibility of the experimenter and is explicitly modelled as a **ParameterSetting** class contained by Experiment.
Input parameters are defined by a **name**, **datatype**, **label** and other properties familiar for example from the PARAM field in VOTable[13]. Most of these are inherited from the **Field** class, which will be discussed 4.6 below.
Because the details of the parameter are defined on the **InputParameter** class, the **ParameterSetting** would not require more than a pointer (the **inputParameter** reference) to the appropriate input parameter and a value. A problem for this model though is what data type to assign to a possible value attribute. We have no knowledge in advance on the data type of the input parameter for which a value is set. This is only known at the instance level, not at

[13] We generalize the ucd attribute on VOTable's PARAM and FIELD to a label attribute with stereotype <<skosconcept>>.

the model level. We do not know whether a certain parameter will be integer, or real, or maybe a string. Our current solution is to allow two different representations of a value, namely a **numericValue**, of type real and a **stringValue**.
This issue and the usability problems it causes will need to be discussed and handled at the IVOA protocol level[14]. One approach is for example the SimTAP idea detailed in the SimDAL document [23].

4.5 Target: Goal of experiment

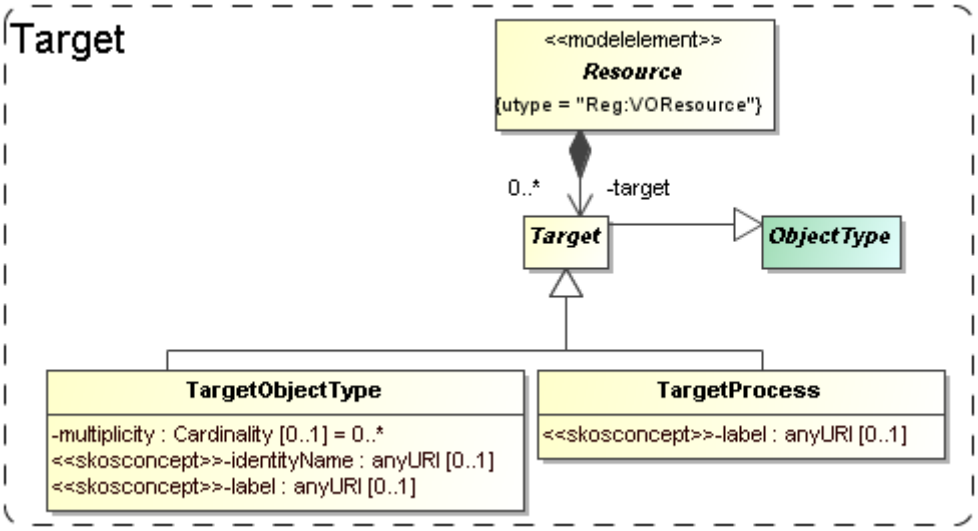


Figure 6 Modelling the goal, or target of a generic resource as objects and/or processes.

Generally the first piece of information that the scientists we polled were interested in regarding simulations was *what* was simulated. I.e. what type of object: a galaxy merger, a galaxy cluster, the large scale structure of the universe? This information in general says something about the goal that the scientists running the simulation had.
In certain cases the simulation code itself may completely prescribe the type of objects and physical processes that are modelled. As example take population synthesis models such as the Galaxev library[15], producing spectra of galaxies.
But many simulation codes allow many different types of objects to be modelled, and even allow one to vary which processes are actually modelled. Also in many cases the actual object that is being simulated is not an intrinsic property of the simulation code, but is a derived property of the actual simulation. For example an N-Body code in general does not contain “galaxy particles”. But one can use it to follow the evolution of millions of low mass particles that are in a particular configuration that together model a galaxy. But it can also be a globular cluster, or a filament in the large scale structure.

[14] Statistical summary has the same problem, see 4.7
[15] Bruzual and Charlot, 2003: http://www.cida.ve/~bruzual/bc2003

To cover the concept of the target of an experiment or protocol, or the goal of a project, we add two classes, **TargetObject** and **TargetProcess**. A TargetObject represents an object, or a physical system in the real world, such as a galaxy, a star etc. TargetProcess represents a physical process such as gravitational clustering or turbulence. This recognises the fact that some simulations are run with the goal of investigating a process, rather than producing a model of a physical system.
Both these classes are subclasses of **Target**, which itself is again a subclass of ObjectType defined in the next section. Target is contained by Resource so that by inheritance they are available to all sub classes. We do not model the Target objects in full detail. For example we rely on a semantic **label** attribute to give a standardised name to the type of object.

4.6 Object types: real and simulated

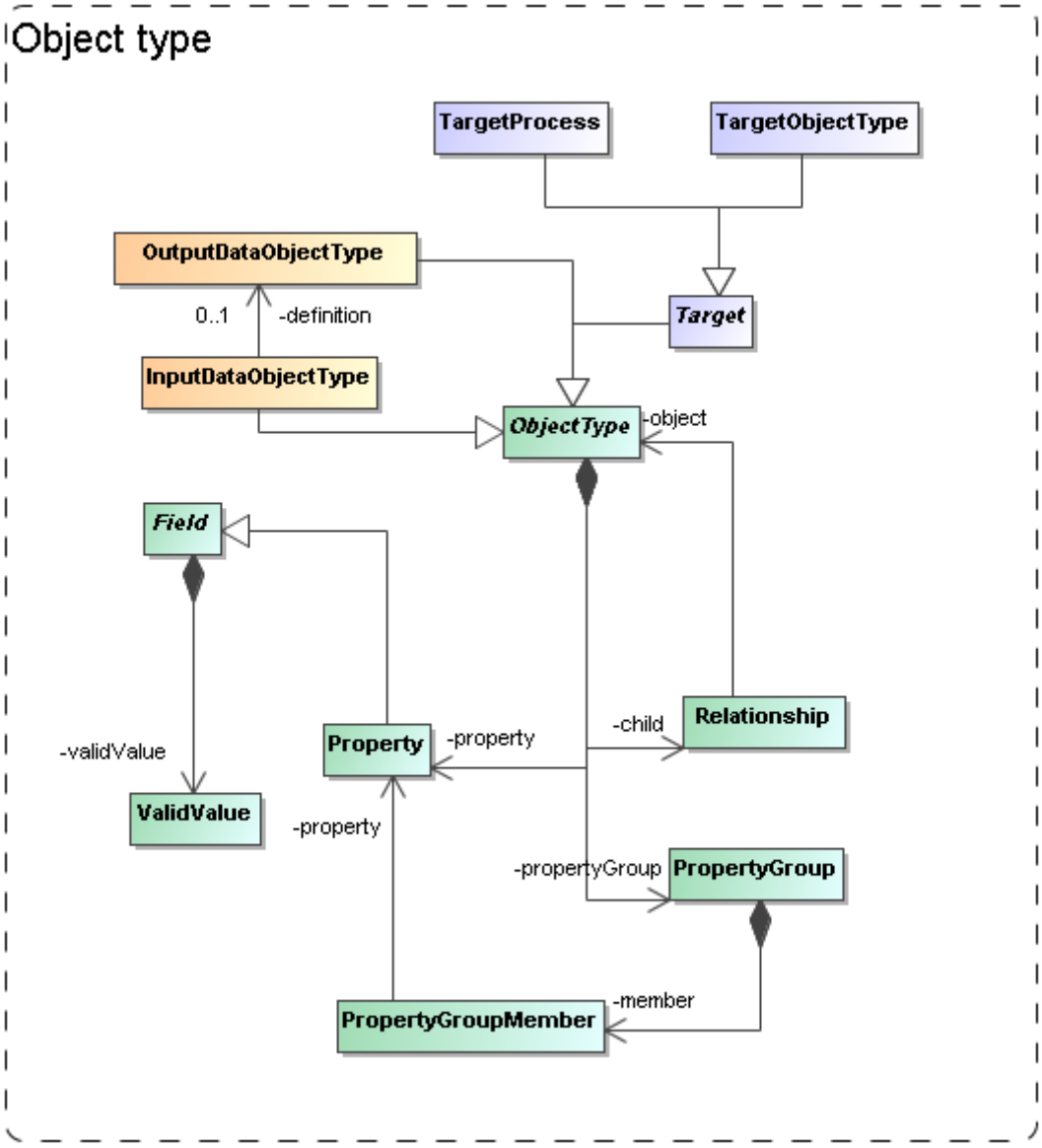


Figure 7 The model needs to describe the types of objects that are being used and produced. We model this in quite some detail in a hierarchy of object types, with properties, grouping of properties and child objects corresponding to nested objects.

In a few places in the model we need to represent the fact that different simulation codes and protocols, or different experiments, need to describe the types of objects they use or produce. For example, the **SimDM/Protocol** must be able to describe the building blocks it uses to represent the "model world" it produces. These building blocks are described by properties and we simply mimic an object oriented design here. That is, we allow users to define object types, with properties and the possibility of relations between parent and child objects.

For this purpose we introduced the **SimDM/OutputDataObjectType** in 4.3 and the **SimDM/Target** (**Object** and **Process**) in 4.5.
We gather the common features of these definitions in the abstract base class **SimDM/ObjectType**. An ObjectType contains a collection of **SimDM/Property**-s that correspond to the simple attributes used to describe an object. SimDM/Property is a subclass of **SimDM/Field** which defines its main attributes such as name, description and data type. Also a SimDM/Protocol's InputParameter *is a* Field, similar to the way a VOTable's PARAM and FIELD share a common structure. Another similarity with the VOTable structure is the possibility to group Property-s in a **SimDM/PropertyGroup**.
To model (hierarchical) relations between different Objects an object has a collection of **SimDM/Relationship**-s, which can represent compositions, aggregation or usage relationship with other SimDM/ObjectType-s.

4.7 *Results: output data and their statistical summary*

We assume users of a Simulation Database will want to gain access to results of simulations and related experiments. This is the same as we assume of users of Simple Image Access or Simple Spectral Access services. For those services the user knows what to expect, a FITS image in one, a spectrum serialised according to the spectrum data model in the other.
Such expectations are not realistic for simulations though. The main problem is that we have no *a priori* knowledge about the contents of their results. Arguably somewhat simplistically one may claim that images and spectra contain pixels with known properties (space, wavelength, flux). Results of simulations, even when constrained to 3+1D simulations, can contain as their *fundamental constituents*: point particles, particles with size and structure, mesh cells of fixed or varying size, Voronoi cells[16], structured halos, galaxies, radiation fields, galaxy merger trees etc. And any of these object types can come with any collection of properties: position, velocity, mass, temperature, chemical composition, entropy etc.

[16] http://www.mpa-garching.mpg.de/~volker/arepo/

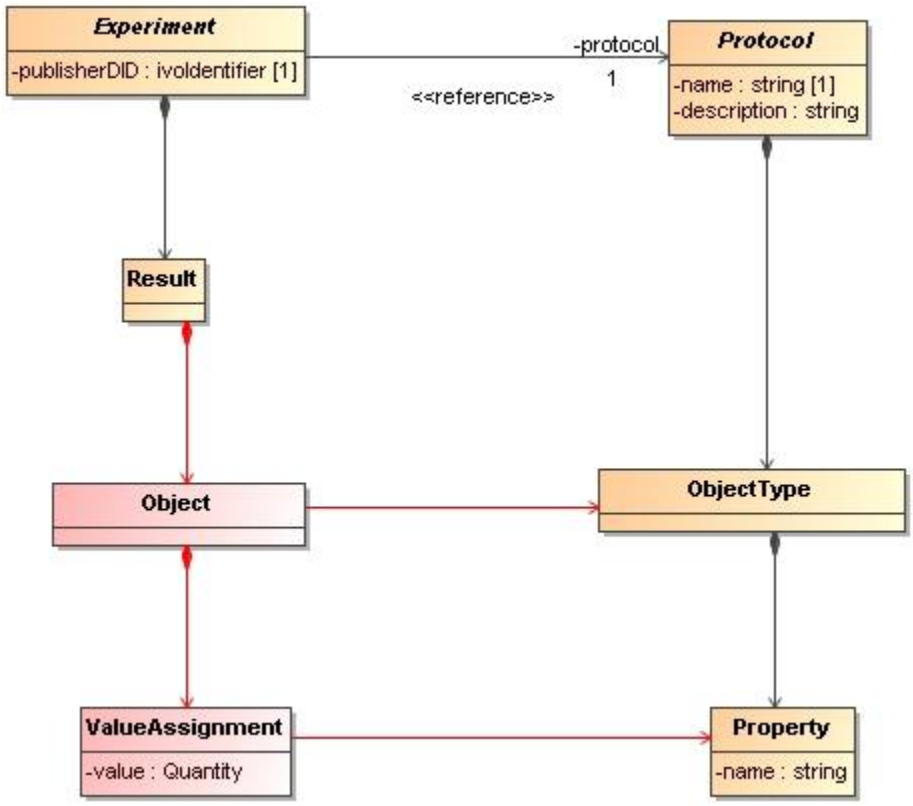


Figure 8 Domain model for results.

Precisely for this reason users will want to have knowledge about the contents of simulation results to decide which simulations might be of interest to them. Hence the model must support description of the results explicitly and Figure 8 illustrates how this is achieved in the domain model:
Experiments produce Results that consist of Objects (pixel, N-Body particle etc) of a particular (Object)Type. The ObjectType defines the structure of Object as a collection of Properties (position, velocity, flux etc), and an Object, being an instance of the ObjectType, assigns values to these properties. Which ObjectTypes and Properties are available is defined by the SimDM/Protocol according to which the Experiment is performed.

The SimDM deviates from the domain model in that it does not include the Object and ValueAssignment classes. Including these would imply that positions, velocities etc for all particles in a simulation are added to the metadata description. For the purposes of SimDB, i.e. discovery of potentially interesting simulations this would be overkill. It is certainly possible to include data as in the domain model, the spectrum data model is a case in point. But that model has a different purpose, namely providing a serialisation of spectra in a standard

manner. Furthermore, individual spectra are quite small and have a well defined structure.

How we model results and their contents in SimDM is shown in Figure 9. Experiments produce one ore more OutputDataSet-s. Such a data set represents one of the possible multiple collections of objects of a single ObjectType represented in a Result. It is in general possible to choose which ObjectTypes from a SimDM/Protocol are included in an Experiment. For example many SPH codes support simulations where only dark matter particles are represented, as well as simulation which include gas and star particles. It is even possible to include some object types only after a certain time.

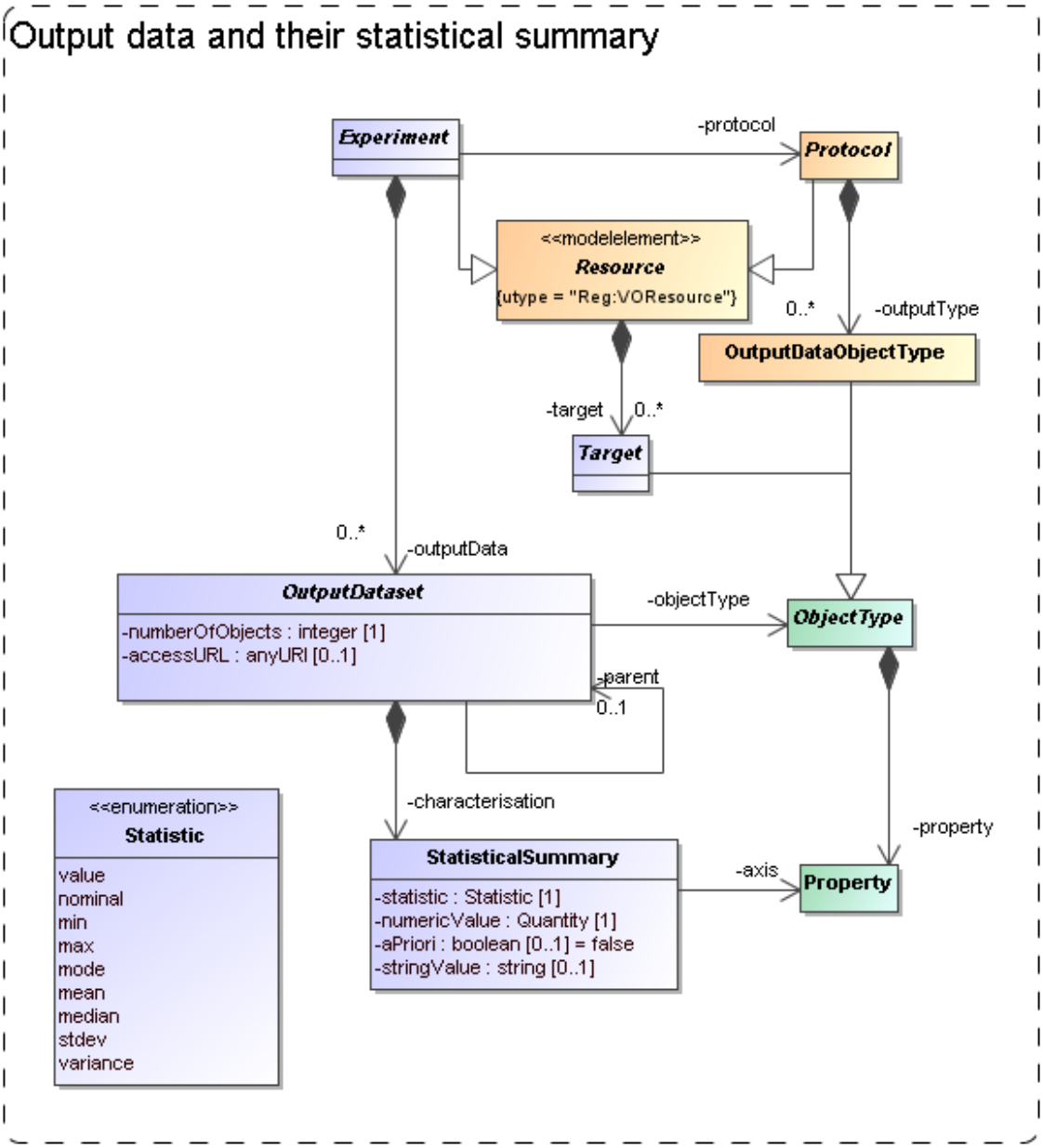


Figure 9 Modelling Results, Products and their contents.

It is in certain cases useful to have some more quantitative information about a simulation. For example, apart from the fact that a simulation has N-Body particles with properties position, velocity and mass, it might be of interest to know that the typical mass of the particles is 10^{10} solar masses.
We support this by allowing users to describe properties of the collections of objects in a Product using a class we call **StatisticalSummary**. This reflects our belief that statistics is the appropriate way to introduce some quantitative aspects of these large collections of objects. StatisticalSummary is contained in Product. It assigns statistical values such as a mean or a min/max value to **Properties** of the **OutputDataSet.objectType**. Which statistic is used is described by the **statistic** attribute.
The data type of the value is a problem, similar to discussed for the value of ParameterSetting
Extensions of this statistical summary to more detailed summaries such as histograms can be easily imagined, but have been left out for the model as they will have less relevance for discovery, which is the main use case for the model.
One further feature is important and pertains to the boolean **aPriori** attribute. This attribute describes whether the statistic that is used in the summary is an *a priori* or an *a posteriori* statistic. An *a posteriori* statistic is calculated using the results after they have been obtained during the running of the experiment. For example an *a posteriori* mean will likely correspond to the usual expression,

$$\frac{1}{N}\sum_{i}^{N} a_i \ ,$$

where the a_i are the values of some property.

In contrast *a priori* statistics characterise the possible values of the observables *before* the experiment is run. In certain cases *a priori* knowledge is available that restricts the possible values that certain properties may obtain in an experiment. An example is a lower bound set on the number of particles that a cluster must contain to be included in the result of a cluster extraction of an N-Body simulation. This can be indicated by a StatisticalSummary object with **statistic**=**min** and **aPriori**=**true**.
Knowledge about the *a priori* statistics is important in the interpretation of the results. In the previous example, when interpreting the mass multiplicity function of a cluster catalogue extracted from an N-Body simulation, it is clearly important to know what the lower limit was on the mass of clusters.
In general *a priori* statistics are the result of, and may often be derived from the input parameters. However this derivation may not be obvious and will in general require intimate knowledge of the parameters of a *(experimental)* protocol. The *a priori* statistic may then facilitate the discovery of catalogues that should contain halos of a certain mass.

4.8 Data access services

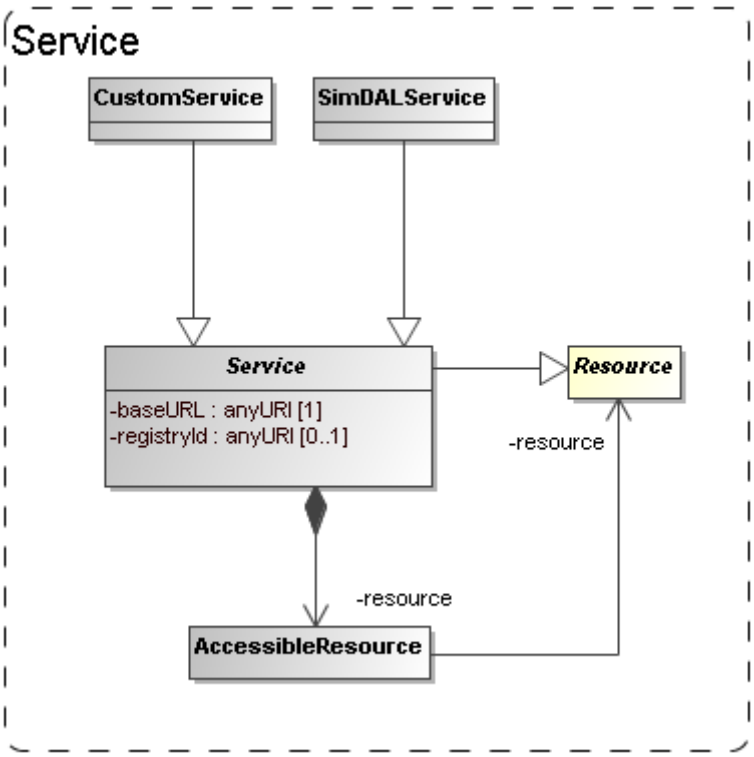


Figure 10 The model for (web) services giving access to SimDM/Resource-s.

The goal of the Simulation Database is to allow scientists to find simulations of possible interest. Once these are found the question is what can be done with them. Clearly knowledge of their existence will be useless if the researcher will not be able to somehow gain access to the results. The usual way this is done in the IVOA, for example in the simple image and spectral access protocols, is that the result of a discovery query contains an access URL that may be used to download the actual image or spectrum, where moreover the format of the returned resource, FITS, VOTable or XML document will be known beforehand.
It was perceived from the beginning of the SNAP project even that for the type of simulations that were supposed to be described a simple download would be unfeasible simply based on the size of many of the typical N-Body or AMR simulations. This assumption still holds and the SimDAL protocol is designed to define special purpose services for retrieving parts of such simulations for example.
Also in the data model we want to indicate how the relation is between the results and services. This part may be used in the SimDB specification to allow users to register services and the Resources they give access to.
In the model the **Service** class, already introduced in 4.2, represents such access services (see Figure 10). This class can be explicitly linked to the Resources it gives access to through a collection of **AccessibleResource**-s.
This part of the model is still rather summarily treated and may need to be updated depending on developments in the SimDAL specification.

5 Serialisations

According to policies of the data modelling working group, first decided in Cambridge, 2003, a data model should be presented using a UML diagram, a corresponding XML schema and a list of UTYPEs. We have created these both using rules that derive the products directly from the XMI serialisation of the UML data model.

5.1 SimDM/UTYPE

The original goal of the data model presented here was to define the structure of a relational database supporting the Simulation Database *service* specification. A first draft of a note proposing that spec can be found in [22]. SimDB will use TAP [10] to define the IVOA protocol for querying this database using ADQL. The results of such queries will be tabular and serialiased as VOTables. Such a VOTable will contain a filtered subset of the information in the database, but in general in a different form compared to the structure of the data model. To indicate the meaning of data elements in such a VOTable, the IVOA has invented the concept of UTYPEs.

A UTYPE is a "pointer into a data model"[17]. The VOTable XML schema implements this concept as attributes on various elements, e.g. FIELD and TABLE and many other elements. The value of such a UTYPE attribute should identify an element in a data model that is represented by the element itself. For example a table might point to a class definition in a data model, and a column (FIELD) to an attribute.

It has become common practice to provide for an IVOA data model a list of UTYPEs. The Spectrum data model (see [11]) was the first to add explicit UTYPE-s for each of the attributes in its model and the Characterisation data model [16]has followed that example. We follow these examples by assigning UTYPE-s explicitly to all elements in the model.

Our goal was not to have to make this a separate effort, but if possible to generate the list of UTYPEs directly from the model. Our goal was to assign UTYPEs to all identifiable elements in our model and these should be unique.

To this end we define a set of production rules phrased using the special names in our UML profile. We have made a guess as to what the format for UTYPEs will be. In the previous data models a UTYPE existed of a word consisting of dot-separated "atoms", similar to UCDs, but without the ";". We use a slightly different format to make the distinction between different syntactic elements from the profile somewhat clearer and also to guarantee uniqueness of each UTYPE within the data model context. Once (if?) a format is settled on within the IVOA we will easily be able to adjust our definitions.

The important point we want to make is that it is possible to define simple rules that can automatically produce *unique* UTYPE-like words for all elements of a

[17] See 6.3.3 for our position on the discussion that is still going on regarding UTYPEs.

data model, i.e. the only discussion that may be required is on the rules for doing so IF a fixed format is preferred (see Norman Gray's ideas[18] on why this might not be necessary).
The following BNF-like expressions define the particular rules we have used for deriving the UTYPEs from the UML model:

```
utype              :=    [model-utype | package-utype | class-utype |
                         attribute-utype | collection-utype |
                         reference-utype | container-utype
model-utype        :=    <model-name>
package-utype      :=    model-utype ":/" package-hierarchy
package-hierarchy  :=    <package-name> "/" [<package-name> "/"]*
class-utype        :=    package-utype <class-name>
attribute-utype    :=    class-utype "." attribute
attribute          :=    [primitive-attr | struct-attr]
primitive-attr     :=    <attribute-name>
struct-attr        :=    <attribute-name> "." attribute
collection-utype   :=    class-utype "." <collection-name>
reference-utype    :=    class-utype "." <reference-name>
container-utype    :=    class-utype "." "CONTAINER"
identifier-utype   :=    class-utype "." "ID"
```

For the SimDM these rules produce a list of UTYPEs for the model. For each model element we provide the UTYPE in the HTML documentation in [6] and we provide a complete list at the end of that document[19]. Note also that a URL of the type

```
<URL-to-HTML-doc>#<utype>
```

will link one directly to the documentation for the corresponding data model element. This is in conformance with a suggestion made by Norman Gray[18].

When representing components of the data model in a VOTable (for example), these UTYPEs SHOULD be used, in particular when the VOTable contains results of ADQL queries to a SimDB/TAP implementation (see *SimDB Services*).. Alternative views and representations of the SimDM, for example in SimDAP, SHOULD use these UTYPEs to refer to elements in the model.

5.2 XML

A specification for an IVOA data model should (must?[20]) contain an XML schema that defines how to serialise data model instances as XML documents. Similar to the case of UTYPEs we did not want to make the design of these schemas a separate effort; instead we want to derive the schema from the model. To do so

[18] http://nxg.me.uk/note/2009/utype-proposals/
[19] http://volute.googlecode.com/svn/trunk/projects/theory/snapdm/specification/html/SimDM.html#utypes
[20] See "Rules" on http://www.ivoa.net/cgi-bin/twiki/bin/view/IVOA/HowToParticipate This "decision" was made in the Cambridge 2003 interoperability meeting together with the requirement that data models must be specified in UML.

we have defined rules for relating XML Schema constructs to our UML model. These rules are a completion of those described in [37]. It is based also on a view of what such schemas should look like, restricting the possible set of constructs to be used in schemas representing data models. These design rules have earlier been discussed with and accepted by the Registry and VOTable working groups.

We give here only a short description of these rules. First of all we define two different types of schemas. First we define “type schemas”, XSD documents containing only type definitions. For each object type(class) and value type we generate a corresponding complexType or simpleType. Attributes map to elements of a corresponding data type (simple or complex), collections to elements of a type corresponding to the class. References are harder to represent and will be discussed below.

We next generate a “document schema” containing root elements. The elements in the document schema define the valid XML documents one can write and we choose only “root-entity classes” for their type. That is, only classes at the root of collection trees can be represented as a document. Fragments of these are not allowed. For example, only a complete Simulator or Simulation can be represented in a document, not only a single result, or parameter setting.

Note that this is a choice made for the Simulation Database service specification. The document schema depends on the type schemas through XML schema import declarations. This separation allows flexible usage of the type schemas, for example other services might make a different choice from the types to serve as valid root elements.

The root schema for the SimDM/XSD representation can be found here[21]. The type schemas and a predefined base schema can be found in the same directory and subdirectories of it. We refer to the *SimDB Services* document for more details on the XML schema serialisation and their use in the SimDB service protocol.

Only the mapping of references deserves special attention. Our choice of mapping from UML to XSD elements and our definition of root elements imply that many references must be able to link between different XML documents. For example the *(experimental)* protocol reference[22] in an XML document describing an Experiment must be able to identify a *(experimental)* Protocol that is defined in a different XML document. To do this identification we assume we must rely on an agent that can interpret a serialisation of a reference and use it to look up a corresponding document. Therefore we map references to elements of a particular complexType that we define in a base schema[23]. That same schema defines a type to be used for representing identifiers of objects and the reference serialisation must be able to reproduce such an identifier.

---

21 http://volute.googlecode.com/svn/trunk/projects/theory/snapdm/specification/xsd/SimDB_root.xsd

22 UTYPE: SimDB:/simdb/experiment.Experiment.protocol or http://volute.googlecode.com/svn/trunk/projects/theory/snapdm/specification/html/SimDB.html#SimDB:simdb/experiment/Experiment.protocol

23 http://volute.googlecode.com/svn/trunk/projects/theory/snapdm/specification/xsd/base.xsd

Further technical details of this mapping will be described in the appropriate service definition document.

6 Dependencies on other IVOA efforts

IVOA documents are assumed to specify dependencies on other IVOA efforts. We have from the beginning realised that the SimDB effort touches upon various other specifications and general efforts of other working groups [22]. Here we discuss these relations as far as they pertain to the Simulation data model.

6.1 Registry

The correspondence between the full Simulation Database specification and the IVOA Registry will be discussed in the *SimDB Service* note [22]. Here we will address the relation between the SimDM and the Registry Data Model as defined in [14].

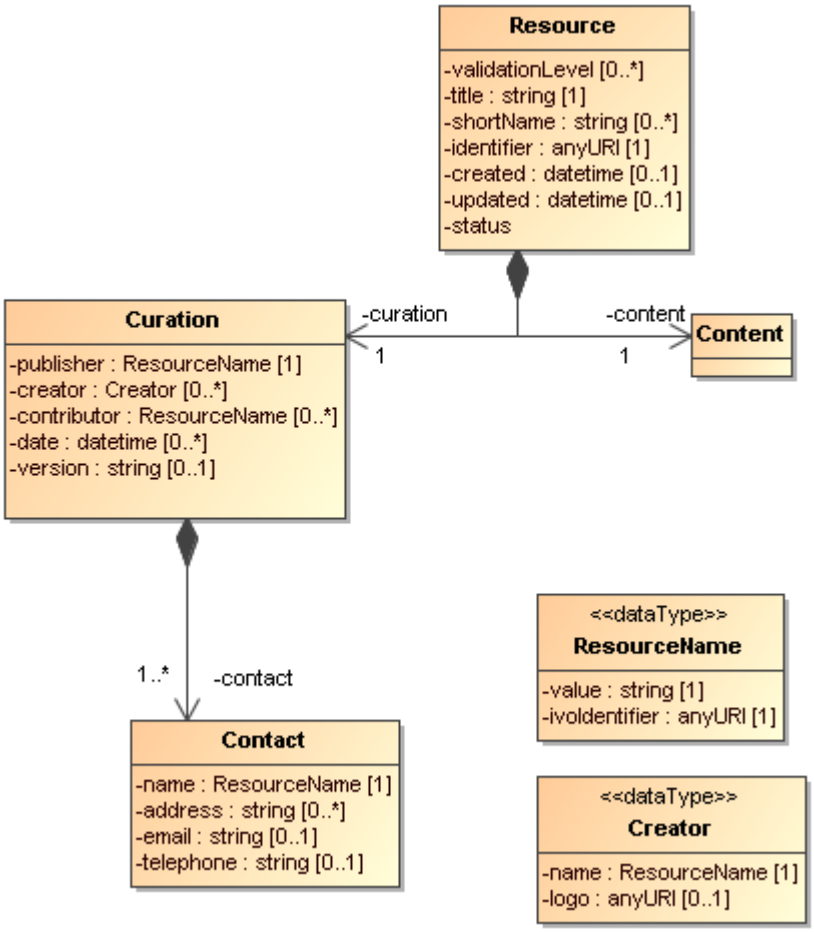


Figure 11 UML rendering of the Resource complexType from [14].

In Figure 11 we present a UML rendering of the *Resource* complexType as inferred from the Resource Registry VOResource XML Schema [14]. Comparing that model to SimDM/Resource we can see that these two models for Resource are related, but not identical. In data modelling terms, it is not true that a SimDM/Resource *is a* Registry/Resource (or *vice versa*). *Curation* is modelled differently and arguably with less detail in SimDM, but the main difference is in the *Content*. SimDM provides a very detailed and specialised model for the *Content* of Simulations and related resources, by modelling provenance, motivation and results characterisation. This higher level of detail gives rise to a higher level of granularity in the types of resources stored in a SimDB, which in general will be to fine grained for registration in a Registry. This is similar to the case of a single image, which is not a Registry/Resource, whereas a SIAP-compatible *service*, providing access to many images, is.
A SimDB service itself will have to be registered (see chapter for that discussion), i.e. a SimDB service *is a* Registry/Resource. In discussion with Ray Plante (IVOA Interoperability meeting May 2007, Beijing) on this issue it was proposed that some part of the contents could also be registered in a Registry directly, i.e. we should be able to identify Registry/Resource-s in SimDB. Considerations to decide on how to make this identification would be for example that all data products resulting from a well-defined (and published) scientific project could qualify. To represent such a possibility for now we have introduced another subclass of SimDM/Resource: SimDM/Project. This is not much more than an annotated aggregation of other SimDM/Resources, with some additional attributes describing the motivation etc. The metadata of a SimDM/Project is not the same as that of a Registry/Resource, however we propose that we should be able to define a transformation (possibly implemented again in XSLT) to transform a SimDM/Project and produce a Registry/XML representation.

6.2 Semantics: Use of SKOS Concepts

In the SimDM, observables, object types, properties, parameters that play a role in a given simulation have to be defined explicitly, for the world of simulations is too large to define all possibilities explicitly in the model itself. This in contrast for example to the spectrum data model [11] where we know that a flux is determined for a wavelength interval, or a model for images where a flux is determined for a spatial pixel. In principle the publisher of a SimDM/Resource has all freedom to name and describe these entities. For other users to understand the meaning of them, we have where appropriate, added an attribute corresponding to a semantic label. This is similar to the situation in VOTable, where FIELD-s can be given a UCD (or UTYPE) that allows users to understand the meaning of a column in the table.
In SimDM we need to generalise this concept as UCDs are not sufficient for our purpose. For example target object types are not covered by the list of UCDs and the same for other elements in our model. The Semantics WG has specified that such vocabularies should follow the SKOS specification [25]. They have also defined a number of such semantic vocabularies in the SKOS format, for

example of astronomical objects. We try to anticipate their results by introducing a special type of attribute in our UML profile that corresponds to a concept in a given ontology.
Technically, in the UML profile we have defined a stereotype <<skosconcept>> that can be assigned to an attribute in the UML model. Attributes with this stereotype must define a value for the tag "broadestSKOSConcept".
The intent of this is as follows (thanks to Norman Gray for providing the original text with this formal definition):
<<skosconcept>> attributes take a skos:Concept as their value. In each case, the value is given as a single skos:Concept: such attributes may take any skos:Concept which is a narrower concept than this single typing concept. To be precise, for a typing concept T, any concept c is a valid value for this property, if either:

c skos:broaderTransitive T

or if there exists a concept X such that

c skos:broaderTransitive X. X skos:broadMatch T

This just means that, if c is in the same vocabulary as T, then it's connected by a chain of any number of skos:broader, and if it's in a different vocabulary, then there is some X which is in the same vocabulary as c, with a cross-vocabulary link between X and T.

In several cases -- particularly those vocabularies which have been created for SimDM -- there will be a single top concept which everything is narrower than. In other vocabularies -- such as the AstroObject in the thesaurus version of the ontology of object types -- the natural typing concept is not a top concept, or is not the only top concept. This definition also does indicate that it's legitimate for concept c to come from a different vocabulary from T: the fact that c has been declared to be narrower than T, either implicitly or explicitly, is to be taken to be the expression of the vocabulary designer's intention that this be a legitimate value for this property.

6.3 Data Model

6.3.1 UML Profile

The data model proposed in this document is fully defined in all detail through a UML model. UML is a large language and we have consciously restricted ourselves to a subset of the possible modelling elements. We have also added a few modelling elements using the extension mechanisms UML provides through stereotypes, tags and predefined data types. This combination of restriction and extensions is referred to as a UML Profile. The details of our profile are described in a separate document [2], added as an Appendix to the current WD.

One reason to put so much emphasis on the UML model is that it allows us to derive various products of this specification automatically. To this end we use the modelling frame work under development in the VO-URP project[24], which is a spin-off of the SimDB effort. Using XSLT scripts developed in VO-URP we can generate HTML documentation (including UTYPE lists) [6], XML schema definitions [7] etc directly from the XMI representation of the UML model.

6.3.2 Characterisation data model

As described in section 4.7, the model allows one to characterise the results of experiments statistically using the StatisticalSummary class. This part of the model addresses similar problems for simulations as does the Characterisation Data Model for observations. We have not followed that model in detail, but have tried to incorporate its main ideas, giving a new interpretation to some of these[25]. We believe the best way to reconcile the two approaches is to see both as specialisations of a more abstract model defining statistical characterisations of data products. A proposal for such a "domain model for characterisation was given in [32].

6.3.3 UTYPE

Section 5.1 describes how we generate UTYPEs for the different elements in our data model. The rules we use to do so have been subsumed in a draft for a Note on UYUPE-s by Mireille Louys [17]. One problem we have with that Note is that the concepts used in the grammar, and that are direct reflections of syntactic modelling elements in our UML profile, have not been defined. For models defined with different UML syntax the grammar does not help.
Some have argued against any semantic meaning to a UTYPE string. It should not be necessary to parse it to find out what its meaning is. Instead one should be able to follow it , but could/should be opaque. It should simply be assigned to the modelling elements. In that case the only requirement would be that a unique list of strings is created and that

Our assumption has been that a UTYPE should allow one to uniquely identify a concept in a data model. We do not assume that our particular form to do so need to be taken over. But, as we describe in 5.1, *if* one wants to simply derive a list of unique strings to be associated to concepts that play a role in data models designed with our UML profile, these rules may help. Clearly if the syntax were to change we can accept that.
The effort on understanding what UTYPEs really are, how they are to be used, or defined is in our opinion not completed. But we feel that our approach is compatible with any possible interpretation, and sufficiently flexible to proposed changes in precise syntax, were they required.

[24] http://code.google.com/p/vo-urp/
[25] This follows ideas presented in China 2007, see
http://www.ivoa.net/internal/IVOA/InterOpMay2007DataModel/CharacterisationInTheDomain.ppt

7 References

7.1 Accompanying documents

[1] This document, at web address http://volute.googlecode.com/svn/trunk/projects/theory/snapdm/specification/PR-SimulationDataModel-v.1.00-20110428.doc

[2] The appendix, at web address http://volute.googlecode.com/svn/trunk/projects/theory/snapdm/specification/SimulationDataModel_Appendix.doc

[3] SimDM UML diagram obtained from MagicDraw : http://volute.googlecode.com/svn/trunk/projects/theory/snapdm/specification/uml/SimDM.xml

[4] A PNG representation of the main diagram, 'all', in the model, extracted form MagicDraw in http://volute.googlecode.com/svn/trunk/projects/theory/snapdm/specification/uml/SimDM.png

[5] "Intermediate representation" of the model. An XML document containing all relevant information from the model in a more readable format than XMI. This document is generated from the XMI and is itself the source of all other generated products. http://volute.googlecode.com/svn/trunk/projects/theory/snapdm/specification/uml/SimDM_INTERMEDIATE.xml[26]

[6] HTML representation of the SimDM in http://volute.googlecode.com/svn/trunk/projects/theory/snapdm/specification/html/SimDM.html

[7] XML schema documents derived from the data model and defining the representation of data model instances in XML. Divided over various documents. The "element schema" document defining all root elements can be found here: http://volute.googlecode.com/svn/trunk/projects/theory/snapdm/specification/xsd/SimDM_root.xsd . All type schemas can be found in the same folder, http://volute.googlecode.com/svn/trunk/projects/theory/snapdm/specification/xsd and sub-folders of it.

7.2 Relevant IVOA documents

[8] Gheller C., Wagner R. *et al*, *Simulation Data Access Protocol (SimDAP)*, http://code.google.com/p/volute/source/browse/trunk/projects/theory/snap/SimDAP.html

[9] R. Hanisch, *IVOA Document Standards*, http://www.ivoa.net/Documents/latest/DocStd.html

[10] TAP specification, we reference the version that was up for RFC: http://www.ivoa.net/Documents/TAP/20090607/WD-TAP-1.0-20090607.html

[11] Jonathan McDowell *et al* (2007) *IVOA Spectral Data Model* http://www.ivoa.net/Documents/latest/SpectrumDM.html

[26] The XML schema file defining the structure of the representation in [5] can be found in the VO-URP project: http://vo-urp.googlecode.com/svn/trunk/xsd/intermediateModel.xsd

[12] *A Unified Domain Model for Astronomy*
Lemson, G., Dowler, P, Banday, A.J., 2004
http://www.aspbooks.org/a/volumes/article_details/?paper_id=861 see also
http://www.ivoa.net/internal/IVOA/IvoaDataModel/DomainModelv0.9.1.doc
[13] Bob Hanisch et al, *Resource metadata for the virtual observatory*
http://www.ivoa.net/Documents/latest/RM.html
[14] Ray Plante et al 2008, *VOResource : an XML Encoding Schema for Resource Metadata*
http://www.ivoa.net/Documents/REC/ReR/VOResource-20080222.html
[15] Carlos Rodrigo et al, *S3 : proposal for a simple protocol to handle theoretical data (microsimulations)*
http://www.ivoa.net/Documents/latest/S3TheoreticalData.html
[16] Mireille Louys et al (2008) *Data Model for Astronomical Data Set Characterisation Version*
http://www.ivoa.net/Documents/latest/CharacterisationDM.html
[17] Mireille Louys *et al* (2009) *Utype : A data model field name convention Version 0.3*
http://www.ivoa.net/internal/IVOA/Utypes/WD-Utypes-0.3-20090522.pdf
[18] Paul Harrison et al, Simple Image Access specification Version 1.0
http://www.ivoa.net/Documents/SIA/
[19] Doug Tody et al, Simple Spectral Access specification version 1.04
http://www.ivoa.net/Documents/latest/SSA.html
[20] Theoretical Spectral Access Protocol
http://www.ivoa.net/cgi-bin/twiki/bin/view/IVOA/IVOATheoryTSAP
[21] Theory in the VO
G. Lemson and J. Colberg (2003)
http://www.ivoa.net/pub/papers/TheoryInTheVO.pdf
[22] Proposal for a Simulation Database Standard, IVOA Note 11 July 2008
G. Lemson et al (2008)
http://www.ivoa.net/Documents/latest/SimDBTrack.html
[23] Simulation Data Access Layer v0.1
Claudio Gheller etal (2010)
https://volute.googlecode.com/svn/trunk/projects/theory/snap/SimDAL-0.1.html
[24] IVOA Astronomical Data Query Language
Inaki Ortiz etal (2008)
http://www.ivoa.net/Documents/latest/ADQL.html
[25] Vocabularies in the Virtual Observatory
S. Derriere etal (2009)
http://www.ivoa.net/Documents/latest/Vocabularies.html

7.3 Other sources

[26] *Framework for the inclusion of theory data and services in the VObs*
Santi Cassisi et al 2008
http://cds.u-strasbg.fr/twikiDCA/pub/EuroVODCA/Deliverables/EuroVO-DCA_D11_MPG_Final.pdf
[27] Martin Fowler (1997) *Analysis Patterns*
Addison Wesley Longman, Inc
[28] Terry Halpin (2001) *Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design*
Morgan Kauffmann Publishers
[29] XML schema, http://www.w3.org/XML/Schema

[30] *MOF 2.0/XMI Mapping, V2.1.1*
http://www.omg.org/spec/XMI/2.1/PDF
[31] *OMG Unified Modeling Language (OMG UML), Infrastructure* Version 2.2
http://www.omg.org/docs/formal/09-02-04.pdf
[32] G.Lemson (2007) *Characterisation in the domain*
Presentation at IVOA interoperability meeting Bejing 2007.
http://www.ivoa.net/internal/IVOA/InterOpMay2007DataModel/CharacterisationInTheDomain.ppt
[33] http://en.wikipedia.org/wiki/Conceptual_data_model
[34] http://en.wikipedia.org/wiki/Logical_data_model
[35] http://en.wikipedia.org/wiki/Physical_data_model
[36] *Data Mining the SDSS SkyServer Database*
J. Gray *etal* 2002
http://www.sdss.jhu.edu/ScienceArchive/pubs/msr-tr-2002-01.pdf
[37] *Model Based Schema*
G. Lemson, 2004. PPT presented during Registry video conference 2004-05-13
http://www.g-vo.org/www/uploads/Documentation/Registry_XSD_videocon20040513-14.ppt