MIRACLE simulation outputs metadata specification version 2.0.0

Gary Polhill, Lorenzo Milazzo, Dawn Parker, Xiongbing Jin, Calvin Pritchard, Ju-Sung Lee, Doug Salt
8 November 2022

Abstract
This document contains metadata specifications for recording simulation outputs.

Contents

ChangeLog

Version	Date	Person	Description
1.1.3	28 April 2015	Lorenzo Milazzo and Gary Polhill	Original version
1.1.4	18 June 2015	Gary Polhill , Lorenzo Milazzo, Dawn Parker, Calvin Pritchard, Xiongbing Jin	Enhancements for workflow specifications and links with a metadata schema from earlier work at the University of Waterloo
1.1.5	7 July 2015	Gary Polhill and Lorenzo Milazzo	Improvements to fine grain, assumptions and statistics; explicit links to possible external ontologies; record of visualisations
1.1.6	26 Nov 2015	Ju-Sung Lee and Gary Polhill	Improvements to the fine grain; cosmetic improvements
1.1.7	29 Apr 2016	Doug Salt	Corrections Add Description field to everything Add About field to everything Remove Purpose from Pipeline as redundant Remove Purpose from Application as redundant. Change statisticalInput, statisticalVariable and statisticalMethod
2.0.0	8 Nov 2022	Doug Salt	Tidied up the relations

Introduction and overview

The schema diagram is shown in Figure 2. There are two important dimensions of distinction. First is fine- versus coarse- grained metadata. Second is provenance versus workflow. Coarse-grained metadata describes how particular files come (or came) into being, or were (or could be) used to bring other files into being. Fine-grained metadata describes specific values recorded in social simulation outputs. To make the distinction concrete, suppose a simulation produces a CSV file. The data within the CSV file are covered by fine-grained metadata, whilst the fact that the simulation produces the CSV file is coarse-grained. Turning to the other dimension, provenance metadata describes what actually happens (run W of simulation X produced output file Y), whilst workflow metadata describes what could happen (simulation X produces an output file of type Z). The distinctions are summarised in Figure 1.

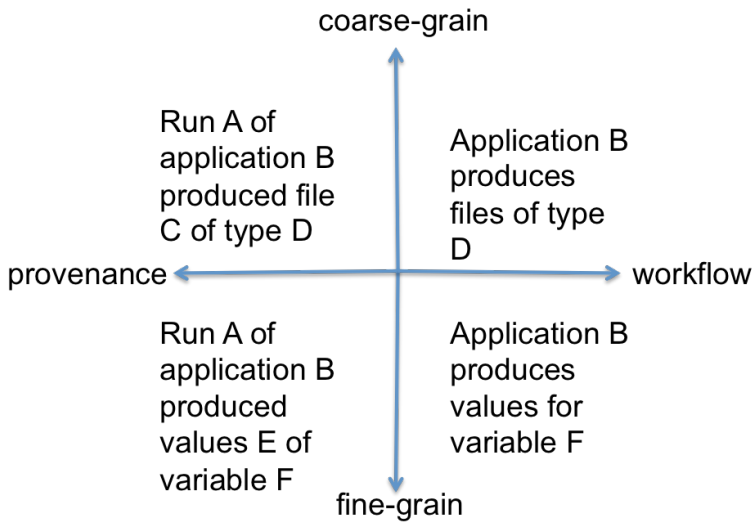

Figure 1.Dimensions of metadata for simulation outputs.

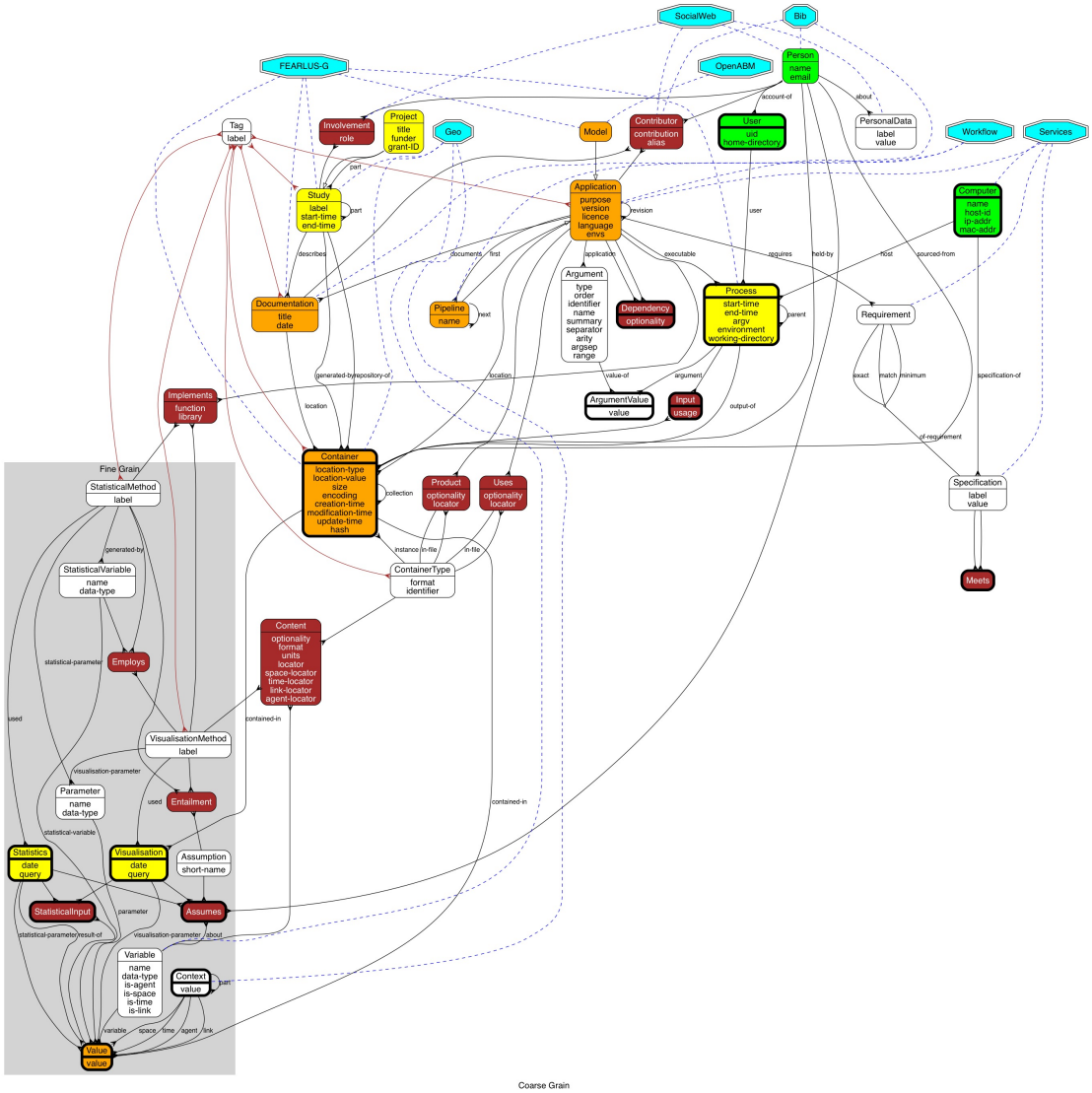

Figure 2. Metadata schema for MIRACLE. Boxes indicate tables, arcs represent relations, with 'forked' arrowheads showing the 'many' part of a many-to-one or many-to-many relationship. Tables are coloured in brown if they are breaking a many-to-many relationship (a reified relationship), with yellow, orange and green being used to denote specialisations from the PROV standard Activity,

Entity and Agent classes respectively. Thick borders denote tables that are potentially autopopulated. Other tables are assumed to be entirely populated by users. Blue octagons show external databases/ontologies to which this one could be linked.

With the schema now having a large number of tables, it is better to consider it in parts. The schema visualisation document is now in a format suitable for using C pre-processor commands to pull out various subgraphs. The following are available:

- ALL: Show the entire schema.
- ANALYSIS: Show the part of the fine grain pertaining only to analysis and visualisation.
- EXTERNAL: Show the links to external ontologies.
- FINEGRAIN: Show everything relating to fine grain metadata.
- FOLKSONOMY: Show the tables that allow tagging.
- PROJECT: Show information relating to metadata about projects.
- PROV: Show tables capturing provenance metadata.
- SERVICES: Show tables pertaining to service-provision and matching requirements against specifications.
- WORKFLOW: Show tables relating to workflow.

Each subgraph can be generated from the schema visualisation document with the (Unix shell) command:

```
cpp -E -Dsubgraph document | grep -v '^#' > subgraph.dot
```

The output file can be read by GraphViz, and exported from there in the required format – e.g.

```
dot -Tjpg -o subgraph.jpg subgraph.dot
```

The overview section now continues with an explanation of each subgraph in turn, with consideration given to the degree of user interaction and maintenance. The remainder of the document thereafter explains each table in detail.

Analysis

The subgraph for recording metadata about Analysis is shown in Figure 3. Variables are metadata about the content of files, and may have several Values. These Values may be visualised, and Statistics may be computed using them. Statistics are computed using StatisticalMethods, and Visualisations constructed using VisualisationMethods. StatisticalMethods produce StatisticalVariables as outputs, which are stored in the Values table as results-of Statistics. Statistics and Visualisations are thus conceived as Activities in the PROV vocabulary.

When a user of the system has identified some statistics or visualisations they want to be able to reuse in other studies or record provenance about, as part of recording the activity, if the statistics are not something already added by another user, they would make an entry in the StatisticalMethods or VisualisationMethods tables. If known, the user would also record any Assumptions Entailed by the methods. This information is not a requirement, as

the user may not have relevant expertise to assert that a particular computation involves an Assumption; however, this information can be added at any time by any user. The Employs table can also be filled with data recording when one Statistical- or Visualisation-Method uses a StatisticalVariable as input.

Assumptions could be quite trivial – in the most simple case, for example, that a numeric Variable is assumed to be a cardinal (as opposed to ordinal or nominal) when computing the mean of its Values. For other statistics, Assumptions can record such things as whether the Variable is normally distributed, or has constant variance.

Once an Assumption has been stated as Entailed by a Statistical- or Visualisation-Method, it can be automatically inferred that Persons have made Assumptions about Variables, where they have done Statistics or Visualisations that use the Methods and Values of those Variables that are returned by the queries used to get the raw data on which the activities operated. The query is stored, along with the date made, in order to avoid populating large relational tables recording the Visualisations and Statistics that used Values as input data. Note that Values may also be used to store Parameters and StatisticalVariables that act to configure Visualisations and Statistics. These are captured using the visualisation-parameter and statistical-parameter relations, and the StatisticalInput table.

Visualisations are automatically inferred when the Container they are contained-in has a ContainerType with a VisualisationMethod in it. The Values visualised (recorded in the VisualisationValue table) can be (possibly somewhat dubiously) inferred from the Input to the Application that ran the Process that created the Container, until such time as statistical tools support logging this information in sufficient detail.

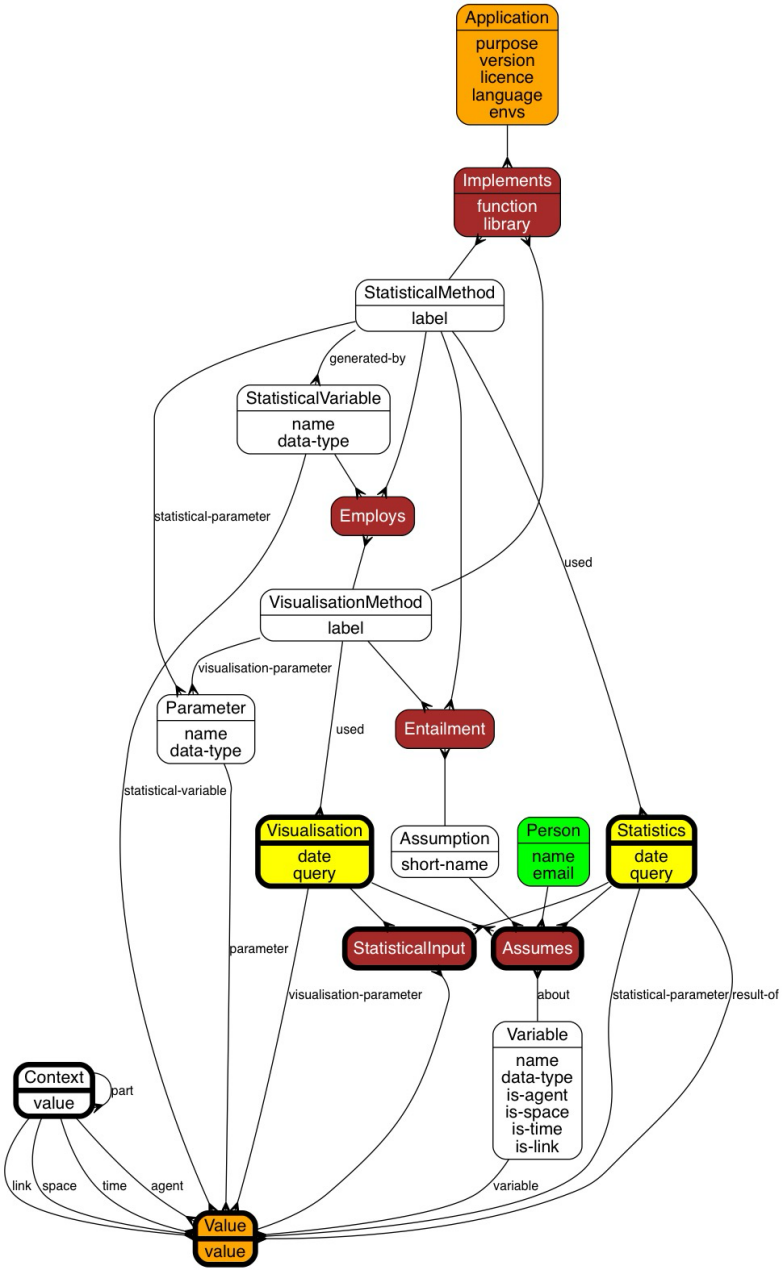

Figure 3. The analysis subgraph of the schema

External ontologies

Though not explicitly noted in the tables, with the exception of OpenABM (with which this schema has to integrate for the MIRACLE project), various external ontologies and schemas are relevant, and could be linked to if required. There are various ways such links could be manifested (relational tables for example, implementing each of the blue dashed lines in Figure 4), and those suggested are by no means exhaustive. Neither is it necessarily the case that any specific ontology or external schema is proposed or adopted. The following gives more specific consideration to each of the proposed external links:

- Bib – connection to a bibliographic database (e.g. bibsonomy). Note that the Documentation table is intended to be quite generic, and include journal and conference articles as well as reports and code documentation.

- FEARLUS-G – links with the FEARLUS-G metadata infrastructure. This was designed for simulation with a specific model, but has concepts that are more generally applicable. In other work, the PolicyGrid ontology also contains relevant concepts, and indeed, concepts from any Virtual Research Environment could be applicable here, though with FEARLUS-G, it was specifically aimed at supporting agent-based simulation.
- Geo – links to ontologies or databases containing geographical or spatial concepts, such as GeoSparql and WGS84. The idea here is that we may want to link various table entries to external geographical databases, for example, to say that a Study pertained to a particular region, or that a Container is a GIS file.
- OpenABM – this is linking back to the CoMSES-Net archive of agent-based models, and is essential in order to identify which model these metadata are describing the output analysis of. (Although in principle, the system described could be applied to any stage of modelling, including setting up the model files, running the model and analysing the output, the focus of the MIRACLE project is specifically on the output analysis.)
- Services – links to vocabularies describing the requirements of applications and capabilities of service-providers. More sophisticated modelling could include facilitating automatic discovery of applications and services, e.g. using WSDL, whilst ontologies such as OWL-S provide some concepts that are also relevant to this schema, including input and output specifications.
- SocialWeb – we may want to allow people to link to social web tools such as ResearchGate, LinkedIn, Facebook and Twitter. The FOAF ontology also has attributes that we can draw on, and includes vocabulary for modelling social web links.
- Workflow – workflow-related ontologies. As yet there is little in the way of standards. SCUFL-2 is under development by the Taverna team, whilst other approaches to modelling workflows, such as Petri-Nets, the Semantic Web Services Ontology and various business process modelling ontologies are all potentially relevant.

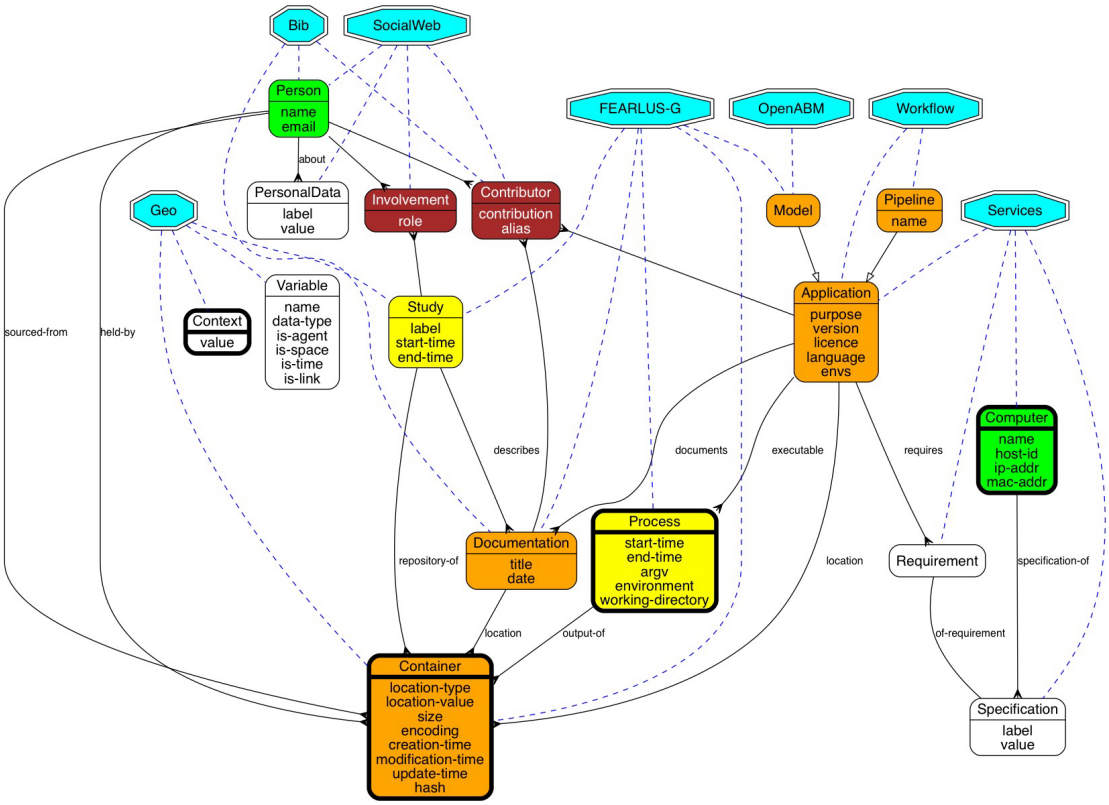

Figure 4. Subgraph showing possible tables that external ontologies could link to, and some relationships between them.

In terms of maintenance, provision for making the links to external databases would require users to supply the relevant information, except where specifically supported databases provided an API allowing us to query them for possibly relevant data to link to.

Fine grain

The fine grain metadata is information about the contents of files, including visualisations, and values of variables (Figure 5). Much of this has been covered already in the Analysis section above, but here we show how Containers are disaggregated from a provenance perspective through saying that Values and Visualisations are contained-in them, and from a workflow perspective through saying that ContainerTypes have Variables and VisualisationMethods as their content. The provenance side can be populated automatically, but the workflow metadata would need to be provided by the user when describing an Application they were making available to the system: specifically, the user will need to describe the inputs and outputs of each application, and for each associated ContainerType, provide details of Variables and VisualisationMethods given as content. For some file formats, autodetection may assist with populating the Variable table – e.g. given an example of a CSV file used by or generated by an application, a header row could be used to suggest names for Variables they supply.

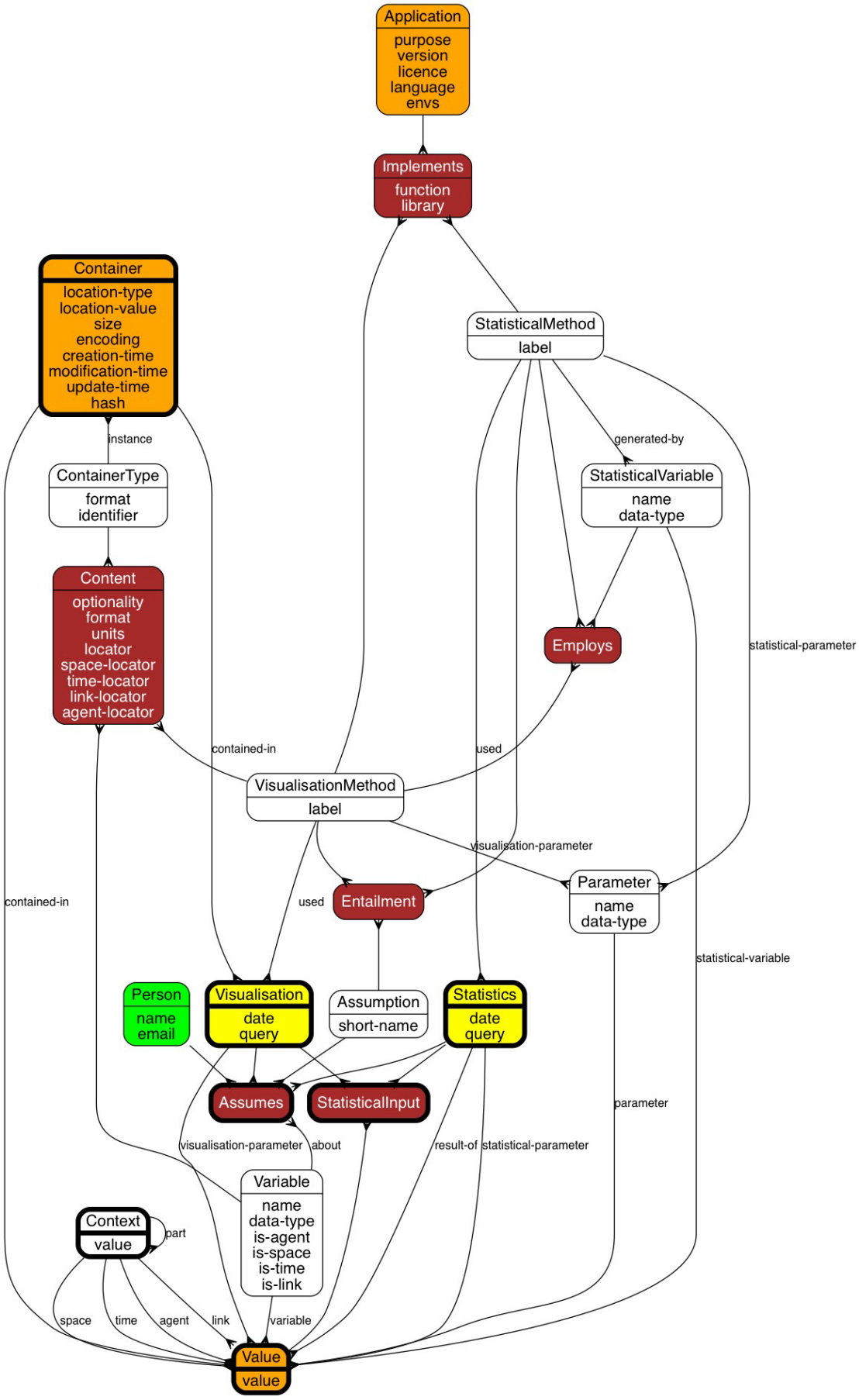

Figure 5. Subgraph showing the fine grain metadata tables

Folksonomy

Folksonomies are informal ontologies developed by a user communities, with tagging being a pretty much standard way in which this is achieved. It is not proposed to enrich that here. In Figure 6, the tables thought most likely to be of interest for users to tag are depicted, though this needn't be exhaustive. Each of the brown arcs from the Tag table would need a relational table to break the jmany-many relationship between tags and the concepts they are applied to.

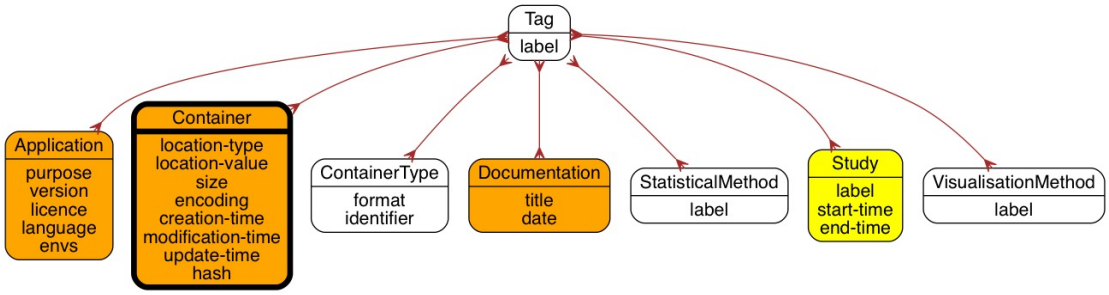

Figure 6. Folksonomy subgraph. Each of the brown arcs would itself be a reified table showing the application of Tags to tables.

Project

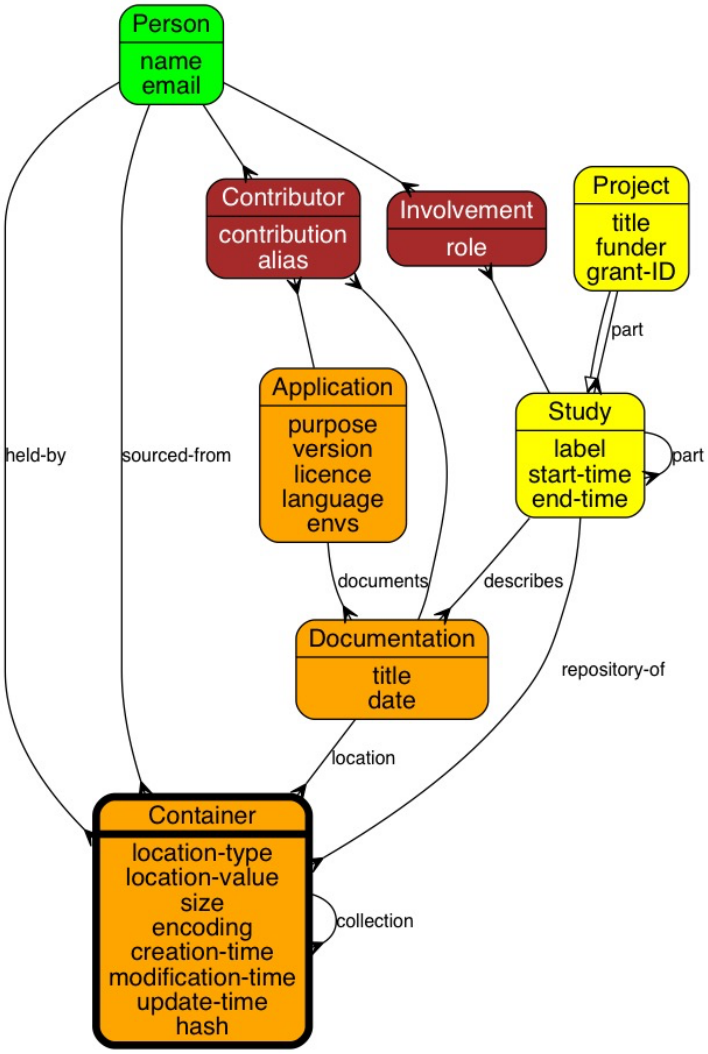

Figure 7. Subgraph capturing metadata about projects

jkkKj:q
Project metadata is largely for users to enter, and hence may be unduly onerous. It is provided to facilitate users in understanding how simulation output data relates to publications (which would appear in the Documentation table), and specific pieces of work (Study table). The Study might be the most important table for users to complete, as this, by its association with Containers allows collection of simulation outputs in useful groups.

Provenance

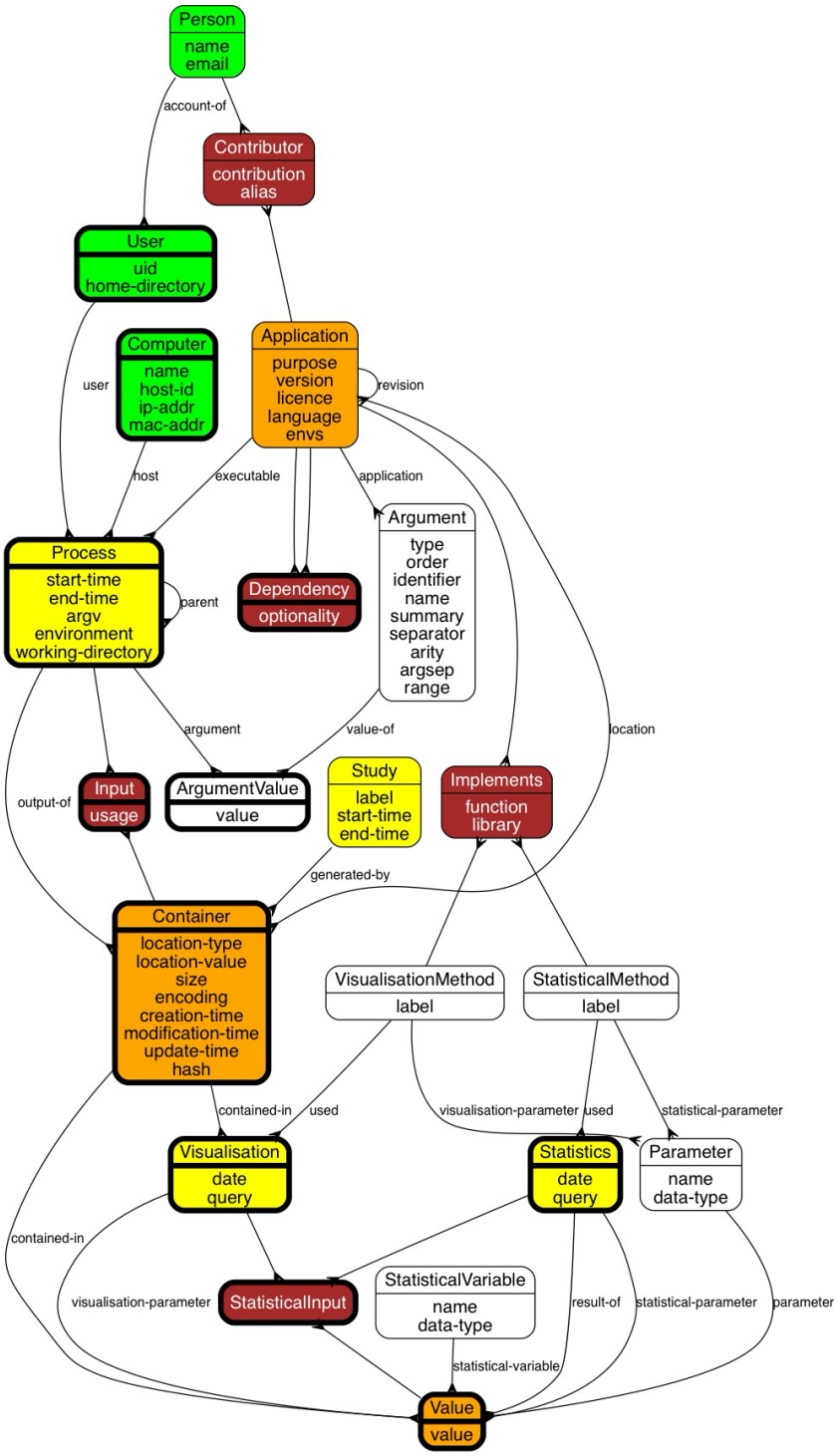

Figure 8. Provenance subgraph.

Provenance is very much at the heart of the system, with an important role in recording how results from a model presented in journal articles are produced. Most of the tables here are autopopulated, as particularly for large scale analyses, user entry is an unrealistic goal. The system therefore needs to act as a wrapper around the scripts and tools (Applications) used to process the raw output from the simulation (which is the starting point of the MIRACLE project) into the result used as part of an article. At the coarse grain, the central activity is the Process, which is a record of all relevant information of a process that ran on a Computer. This includes anything needed to replicate that Process under the same circumstances: the input files it used, command-line arguments, environment variables (essentially, anything that might affect its behaviour), and the output files it generated.

The need for cross-platform support means that it is difficult to draw on standards for these. Different operating systems, and even different commands within the same operating systems, use different conventions for processing input on the command line, and input and output redirection. The Argument table attempts to record all relevant details about command-line arguments, allowing ArgumentValues to be inferred from a specific command-line instruction activating a Process. This is important in avoiding the need for users to supply too much information each time they run an Application.

Batch-file processing is assumed at the coarse grain; where processes involve user interaction that might affect the behaviour, automatic capture of provenance data would be extremely challenging. Whilst GUIs might be used during exploratory analysis of simulation output data, ultimately for reproducibility of results and keeping records of activities done, these must ultimately be manifested as scripts that can be run in batch mode, once a decision is taken that a particular analysis step needs to be recorded as an application. Applications that only support GUI interaction can therefore not be supported (and arguably should be derided as virtually useless for any scientific endeavour – assuming repeatability is a desirable attribute of that work).

At the fine-grain, Statistics and Visualisations are the central activities. Again, although GUI interaction cannot be supported, command-line interaction potentially could. Parameters and StatisticalInput would need to be recorded, and the raw data on which the activities operate captured somehow in a query allowing the same set of data to be operated on. Recording the date at which the query was made is therefore important, especially if Values of Variables are subsequently populated that might affect the ability of future analyses to reuse the same data. A short-cut might be to store the date a Value was generated in the Values table, but since this table is expected to be ‘virtual’ (in that a supporting system would gather relevant data from the raw data files), the date can be captured from the dates of the Processes generating the Containers in which the Values appear, or the date at which the Statistics were computed if the Value is a StatisticalInput.

Services

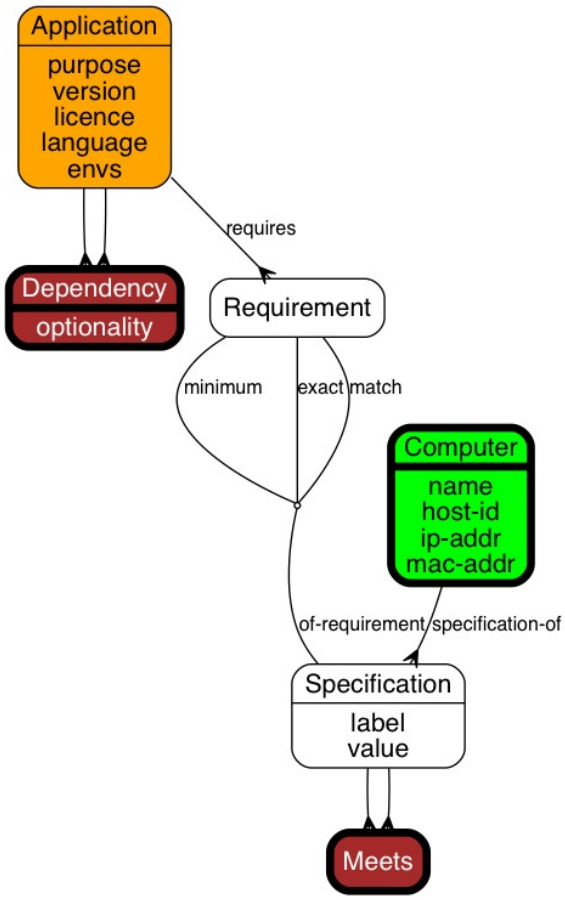

Figure 9. Services subgraph.

TODO: I think the above diagram may be wrong, in that meets is a many-to-many between computer and specification. No it is right but the tables below are wrong. What I think this is trying to say is a Specification is owned by a computer or a specification is owned by a requirement. A specification that is owned by a computer meets a specification owned by a requirement. Requirement is a many to many box and is coloured wrongly.

The services part of the graph depicted in Figure 9 provides a simple model intended to be used to determine whether an Application can be run on a Computer. This involves checking the Requirement Specifications of the Application (and recursively of any Dependencies) against the Specifications of a Computer. Requirements can be specified in one of three ways – for numeric Requirement Specifications, a 'minimum' Requirement must be exceeded by the Specification of the Computer (e.g. for RAM). For all types of Requirement Specifications, an 'exact' Specification must be equal (an example might be the OS – if the Application has very specific demands), whilst a 'match' Specification is a regular expression that the Specification of the Computer must match. The results are stored in the Meets reified relationship.

A more sophisticated implementation would wrap each Application in a web service. This would also be more secure if responsibility for providing Applications as web services was in the hands of their developers – we would then not be uploading Applications to our system, but instead sending data for

processing by other servers, and capturing metadata about these interactions. The services architecture could then draw much more heavily on standard web services ontologies, such as OWL-S and WSDL.

Workflow

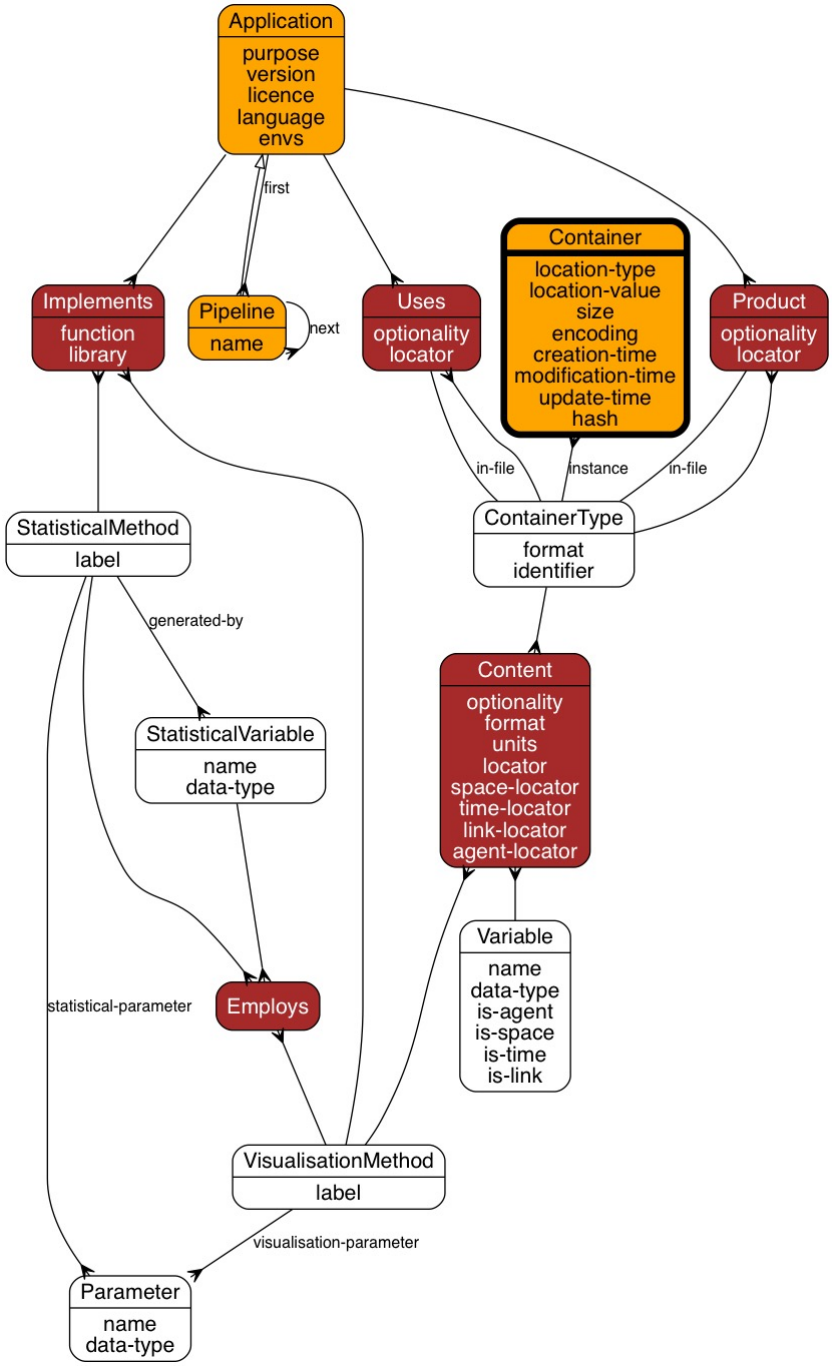

Figure 10. Workflow subgraph

The standard workflow model provides prescriptive 'recipes' for undertaking specific procedures. Though we are interested in that here, and have provided a rudimentary sequence-based Pipeline table to handle that, we are also interested in providing support for exploring what 'could be done' given a current set of circumstances, and 'what needs to be done' before running an Application. For example, suppose we are interested in getting a Value for a particular Variable.

Then we can see which ContainerTypes have that Variable in their Content, and which Applications have the ContainerType as a Product. If the ContainerTypes those Applications Use have no instances, then we can find Applications that have the Uses ContainerTypes as their Product, and so on. We can also search for Pipelines that have the required Product by exploring the Products of the last element of each Pipeline's list-based structure, and checking the Uses of the first element.

This way, it should be possible, given a set of raw simulation outputs, to search for sequences of Applications to apply that generate a particular desired Visualisation(Method). Using the Provenance infrastructure, it would also be possible to explore how other users have chosen to do it in the past.

Detailed specifications

All tables have uniquely specified ID fields as primary keys, unless they are associative tables. All tables with have a name field. This is free form text and not always present. This identifies a relation, however this is not guaranteed to be unique or even present. It is there to help primarily in readability when trying to extract information from this database and may contain a human readable label for the relation. For instance, for the definition of an argument, this would be the argument name. All tables will have a description field (dc:description). This field has been included to allow the introduction of free form documentary text. It is strongly advised that should the opportunity arise, then such text is supplied, even though this will largely be a manual exercise. The authors recommend this practice as it forms a useful means of documenting job control code in its own right, used to run these large scale models. Each table also has a creator/modifier field, and a corresponding created and modified date time field. Although most databases record this kind of metadata, we have made it explicit and expect these fields to be populated automatically. This a deliberate attempt to make this interface database software agnostic. Note that date fields should be stored as strings conforming to the ISO 8601 date format, not in the native format of the database. This is recommended best practice according to the Dublin Core Metadata Initiative.[1] The remaining attribute that is always included in each relation is the "about" column. This we have pinched directly from RDF and uniquely identifies a triple in RDF [cite]. Although currently optional, we intend this to uniquely identify the relationship within our database. The form of this normally some kind of IRI. This will allow inward referencing of the provenance and metadata resource from other standardised resource software that recognizes IRIs.

name	description	A documentary name for the relation
	standards	None
	validation	None. May or may not be present
	automation	None

[1] See http://dublincore.org/documents/2012/06/14/dcmi-terms/#elements-date

description	description	Short text to use to summarise what the argument is
	standards	None
	validation	String
	automation	None

about	description	A unique resource identifier. This field will allow inward linking from any external resource that allow IRIs
	standards	RFC 3987 and RFC 4622
	validation	IRI
	automation	None

creator	description	Who created this particular set of attributes making up this relation
	standards	dc:creator
	validation	String
	automation	Should be done by the agency that has implemented the SSREPI interface. This will be the computer user

created	description	When this document was created
	standards	dc:created, ISO8601
	validation	logical date and time validation
	automation	Should be done by the agency that has implemented the SSREPI interface. This will be the computer user

modifier	description	Who created this particular set of attributes making up this relation
	standards	dc:creator
	validation	String
	automation	Should be done by the agency that has implemented the SSREPI interface. This will be the computer user

modified	description	When this document was created
	standards	dc:modified, ISO8601
	validation	Logical date and time validation. Must come after the date of creation.
	automation	Should be done by the agency that has implemented the SSREPI interface. This will be the computer user

Annotation

This is not shown in the above diagrams. This is database entry that can annotate any relation in this database. This can be used to flag certain properties, or record information about a given set of rows. Such properties will be recorded in a JSON format for the purposes validation.

Standards

JSON for the annotation values. SQL for row restriction.

Automation

There is an entry which documents a database query (if suitable) that will document the rows affected. It is up to the standard implementor whether such queries will be allowed to be performed automatically. We are recommending only statements that refer to the Container table should be allowed to stop the possibility of SQL injections.

Attributes

id_annotation	description	Unique key
	standards	None
	validation	Must be unique
	automation	Automated by the instantiating framework

json	description	json describing the values of the records being annotated.
	standards	None
	validation	Must be unique
	automation	Automated by the instantiating framework

sql	description	The SQL describing the range of records affected. This could be a single rows from single or multiple tables. BEWARE OF SQL CODE INJECTION when considering using this field
	standards	None
	validation	Must be unique
	automation	Automated by the instantiating framework

Relationships

None.

Application

Description

An Application is something that can be run by the user to generate or analyse simulation output.

Standards

PROV:Entity. Note the potential confusion. An Application is something that has the potential to be an Activity (in the PROV sense) in the form of a Process. However, PROV only deals with the past, not with potential. The Application is a file somewhere, and hence an Entity.

Automation

Most of this table is expected to be populated by the user.

Attributes

id_application	description	Unique key
	standards	None
	validation	Must be unique
	automation	Automated by the instantiating framework

purpose	description	Description of the purpose of the application
	standards	?
	validation	Free text
	automation	None

version	description	Version ID for the application
	standards	dc:?
	validation	String (e.g. CHAR(32)).
	automation	

licence	description	Software licence for the application.
	standards	dc:license
	validation	Free text with option to select from other entries
	automation	None

language	description	Programming language the application was written in
	standards	?
	validation	Free text with option to select from other entries
	automation	None

separator	description	The separator used when supplying arguments to the application. This cannot be set at Argument level because this is specific to the application. Normally this would be white space, but it is conceivable

		that this may strictly be a space, or a tab, or a carriage return.
	standards	None
	validation	String
	automation	None

envs	description	List of environment variables to record if the application is run.
	standards	None
	validation	List of strings.
	automation	None

Relationships

revision	description	ID in Application table of Application this Application is a revision of, if any.
	standards	PROV:wasDerivedFrom; dc:isVersionOf
	validation	Must be an ID of an Application
	null	If not a revision of another Application
	automation	None

model	description	ID in Model table of Model this application is, if it is a Model
	standards	None
	validation	Must be an ID of a Model
	null	If this Application is not a Model
	automation	None

location	description	ID in the Container table to use to find this application.
	standards	?
	validation	Must be an ID of a Container
	null	If the search for a Container for this Application has not been done.
	automation	This should be populated automatically the first time the Application is requested on a host by finding the most local Container that references it, and storing that here. If the most local Container is not on the current host, then the Application should be downloaded

		to the current host, and a new Container created for the location. If the Container ID is no longer present, then the search should be repeated in the Container table.

Argument

Description

A command-line argument accepted by an Application. Commands vary hugely in how they parse arguments on the command line, and this table needs to make clear how to build a command line that the Application can use. To be clear, a command line is a string of text that is given to a shell (DOS, bash, etc.) to initiate a batch job.

Standards

POSIX.1-2008 and equivalently IEEE Std 1003.1-2008/Cor 1-2013 are normative for Unix environments, but cannot be assumed even for Unix environments. Cross-platform support is in any case needed.

Automation

When building up a command-line for an Application with arguments during invocation of a Process, the system should do the following:

- Let M be a map from integer to list of string.
- For each flag, check with the user to see whether it should be set or cleared; if set, add the flag name to M, using the order as the key, or -1 if the order is null. Create an entry in ArgumentValue using 'true' or 'false' as the value according to whether or not the flag is set.
- For each option, check with the user to see whether it should be used, and if so, provide an appropriate argument or arguments in accordance with the arity. Build a string for the option including its name and arguments, bearing in mind the separator and argsep values (e.g. for a 2-arity Argument, the string should be a concatenation of name, separator, arg1, argsep and arg2). Add the resulting string to M, using the order as the key, or -1 if the order is null. Use the string as the entry in ArgumentValue.
- For each required argument, request a value (or values) for it from the user in accordance with the arity. Build a string for the values using argsep to separate them if the arity is more than 1. (If a value is given for 'name' (which would be unexpected in a POSIX-conforming command), then put a concatenation of name and separator at the beginning of the string.) Add the resulting string to M, using the order as the key, or -1 if the order is null. Use the string as the entry in ArgumentValue.
- Let K be a list containing a sort of the keys of M.
- Let C be a string.
- For each element of K:
    - Let L be the lookup of that key in M.
    - Let C be the concatenation of C with a join of L (using space as the separator)

- C contains the command-line argument string, to be saved in concatenation with the full path to the command (from its Container) as the argv of the Process.

Attributes

id_argument	description	Unique key
	standards	None
	validation	Must be unique
	automation	Automated by the instantiating framework

type	description	The kind of command line argument this is. Required arguments must be provided, typically don't have a name, and are identified by their number. Options typically have a name, which if given stipulates some sort of value must be provided. Flags are options with arity 0.
	standards	None
	validation	One of "required", "option", or "flag"
	automation	None

order_value	description	A number used to indicate any order in which this Argument should appear in relation to other Arguments the Application accepts.
	standards	None
	validation	Non-negative integer or null if the order is unimportant.
	automation	None

assignment_operator	description	Separator to use between argument name and value. (e.g. space, '=' or empty string (null))
	standards	None
	validation	String
	automation	None

identifier	description	A number or short identifying string for required arguments to identify them
	standards	None
	validation	String. Null if the argument is not required.

	automation	None

name	description	The name of the argument, including any grammar to indicate on the command-line that it is an argument (such as, by convention, / on DOS systems, or –– or – on Unix systems). So, for grep, an example entry might be "––count"
	standards	None
	validation	String
	automation	None

description	description	Short text to use to summarise what the argument is
	standards	None
	validation	String
	automation	None

separator	description	This is the character that indicates that is the argument name. For instance in Unix/Linux this an example might be "--input". This has to be specified as in the case of *nix systems this could be "-input" or "--input". Indeed in old DOS systems this could be "/input". The separator between arguments is set in the Application relation.
	standards	None
	validation	String
	automation	None

arity	description	Number of arguments expected. Can be a (string parseable as a) non-negative integer, ? if 0 or 1 arguments are expected, + if 1 or more are expected, or * if 0 or more are expected. An application may have at most one unnamed argument with arity ?, + or *. If the arity is (possibly) more than 1, then it is assumed that there is no order to these arguments, and that they all have the same range.
	standards	None
	validation	?, + or *, or string containing a non-negative integer. All Arguments of

		type “flag” must have arity 0. No Argument of type “required” or “option” may have arity 0.
	automation	None

argsep	description	Separator to use between arguments if arity > 1 (e.g. comma, semicolon, space)
	standards	None
	validation	String
	automation	None

range	description	Description of the range of any value to be supplied by the user for this argument.
	standards	None
	validation	String
	automation	None

Relationships

application	description	The application this argument applies to
	standards	None
	validation	ID in the application table
	null	Not null
	automation	None

ArgumentValue

Description

A value supplied for a command-line argument in a run of an Application.

Standards

None

Automation

Created when the process is initiated. See Automation under Argument.

Attributes

has_value	description	Value for the argument, or the string ‘true’ if the argument is of type “flag” and the flag is set, or the string ‘false’ if the argument is of type “flag” and the flag is not set.
	standards	None
	validation	None
	automation	Populated when the Process is invoked.

Relationships

for_process	description	Process this ArgumentValue is an argument value for
	standards	None
	validation	ID in the Process table
	null	Not null
	automation	Populated when the Process is invoked.

for_argument	description	Argument this ArgumentValue is an argument value for
	standards	None
	validation	ID in the Argument table
	null	Not null
	automation	Populated when the Process is invoked.

Assumes

Description
Reified trinary relationship asserting that a Person makes an Assumption about a Variable.

Standards
None.

Automation
If there exists a StatisticalMethod *S* that entails an Assumption *A*, and a Statistics was done that used *S*, and a Value *W* appeared in the results of the query used to get the raw data used as input to *S*, and *W* is a value of a Variable *V* then the Person who did the Statistics has made *A* about *V*.

The same can be asserted for VisualisationMethods as for StatisticalMethods.

Attributes
None. (No description.)

Relationships

person	description	Person who made the assumption
	standards	None
	validation	ID in the Person table
	null	If, for any reason, the Person who made the assumption cannot be identified.
	automation	The Person could be the user who ran the Process that computed the Value for the aggregate Variable.

statistics	description	Statistics done causing the assumption to be made
	standards	None
	validation	ID in the Statistics table
	null	Null if and only if visualisation not null
	automation	The performance of the Statistics should trigger the creation of an Assumes entry.

visualisation	description	Visualisation done causing the assumption to be made
	standards	None
	validation	ID in the Visualisation table table
	null	Null if and only if statistics not null
	automation	The performance of the Visualisation should trigger the creation of an Assumes entry.

variable	description	Variable the assumption is made about
	standards	None
	validation	ID in the Variable table
	null	Not null
	automation	This will be identifiable by a query using the Aggregation table.

assumption	description	Assumption made
	standards	None
	validation	ID in the Assumption table
	null	Not null
	automation	Identifiable by a query using the Aggregation and Entailment tables.

Assumption

Description

An Assumption is a condition applied to a Variable for its proper application to a Statistic. (That Statistic being realised as an Aggregation of the Variable to which the Assumption applies.) A Person makes an Assumption about a Variable (in the Assumes table), explicitly or implicitly, every time they compute the Statistic on it.

Standards

None

Automation

None

Attributes

id_assumption	description	A short name for the assumption
	standards	None
	validation	unique
	automation	None

Relationships
None

Computer

Description
A Computer is a machine that runs a Process.

Standards
PROV:Agent

Automation
The details of the Computer should be populated in the system using standard OS calls, whenever a Process is created on a Computer not in the database.

Attributes

id_computer	description	Unique key
	standards	None
	validation	Must be unique
	automation	Automated by the instantiating framework

host-id	description	Host-ID of the computer, if available
	standards	?
	validation	Free text of limited length (e.g. CHAR(64))
	automation	On some unix machines, the value returned by gethostid() (in unistd.h typically), as a hex string.

ip-addr	description	IP address of the computer, if available. Only really of use for computers with a permanent IP address
	standards	?
	validation	Free text of limited length
	automation	ipconfig getifaddr eth0, or otherwise from output of ifconfig

mac-addr	description	MAC address of the computers Ethernet card.
	standards	?
	validation	Free text of limited length

	automation	Can be obtained by parsing output from ifconfig.

Relationships
None

Container

Description
A Container is anything that contains data of any kind that can be accessed from a computer. Most commonly, we anticipate it will be a file, but it could be a folder, archive of files, URI, database, table within a database, etc. Containers are used to provide locations for Documentation and Applications, as well as data.

The Container is the central table for storing coarse-grain provenance metadata about entities involved in creating and analysing simulation output.

Standards
PROV:Entity

Automation
Ideally, if a Process accesses a file (or database table, etc.) as input, there will be a Container for that file and appropriate entry in the Input table. If a file is created as the output of a Process, it will have an entry in the Container table. The Application the Process is running will also have a Container.

From a provenance perspective, we have to allow for the possibility that Containers are mutable, which poses a problem. For example, an experiment might run yielding data that are analysed, but subsequently overwritten by another experiment. When trying to repeat the analysis therefore, we may find that information stored in the Container table is out-of-date. Unless fine-grained provenance data have been stored, the contents of the file will no longer be accessible, though we will still be able to show a provenance graph of how the analysis was conducted. Equally, an Application may be edited after an analysis is done, fixing a bug. Ideally, this would result in a new version of the Application, with a different Container, but we cannot necessarily trust that this will be so unless we archive all data and applications in a pipeline.

Hashing can be used to support tests for equality of Container, though the law of the excluded middle cannot apply (that is, the result of a test of equality could be 'true', 'false' or 'unknown'). If two Containers have different hashes, then they are definitely not the same unless they have different encodings (the charset parameter being of particular relevance for text files). However, if two Containers have the same hash, then they *may* be the same, but are not necessarily the same; if they also have the same size and encoding, it is probably reasonable to assume that they are the same. The location-value, creation-date and modification-date should not be trusted to infer that two Containers are the same.

Attributes

id_container	description	A unique identifier within the database for this container
	standards	None.
	validation	Unique primary key
	automation	Generated automatically by the software framework at time of instantiation

location-type	description	The type of Container. Flexibility should be provided for to allow location types not part of the initial version.
	standards	?
	validation	file, URI, database, remote file, table in database, but with flexibility to add further options.
	automation	For each location type, software will be needed to enable access to it.

location-value	description	Information needed to access the data (where to find it).
	standards	URI? IRI? I assume there is a standard to express all accessible resources, including tables in databases.
	validation	Format will depend on software written to access data for the location-type
	automation	The data will be accessed by subclasses for each location-type that parse the location-value and return the data to the requesting object.

size	description	An *estimated* size for the data in the container in bytes. For folders, the sum of size of all files in the folder and subfolders. Particularly for remote data, and 'volatile' Containers, the size will be that when the container was last accessed.
	standards	None, but note that under ISO/IEC 80000-13:2008, when reporting sizes to the user, the SI prefixes K, M, G, T, etc. should be used for 10^{3x},

		whilst Ki, Mi, Gi, Ti, etc. should be used for 2^{10y}.
	validation	128 bit unsigned integer or equivalent. NULL if the size information cannot be quickly or easily accessed.
	automation	For plain files, most operating systems provide information on the size of the file via API calls (e.g. stat() on Unix systems). Under the HTTP standard, the Content-Length header can be used to get the size of the remote file without downloading it using a headers-only (HEAD) request, but it is not guaranteed to work.

encoding	description	Information on the encoding used for data in this file. MIME types should be used, with subtypes for bespoke user text files. Since this information will not always be available, the encoding should not necessarily be used to prevent users from selecting containers as input for processes.
	standards	MIME
	validation	*type*/*subtype*; *parameter* (as per standard), but note that specific non-standard subtypes (and even sub-subtypes for text/csv) may be needed for bespoke user files. e.g. 'text/fearlus-report-1.1.5; charset=us-ascii' might be used for a FEARLUS report file from version 1.1.5. NULL may be used if the encoding is not relevant for any reason.
	automation	This information is contained in the Content-Type HTTP header. For plain files, it can be accessed in Unix using the `file -I` command. (The –I option outputs in mime type format.) Note that for user-generated bespoke format files, this information will not be provided by standard tools. Metadata about the application that generates the file from the Container-Type (format)

		and Application tables should be used to populate this field when the Container entry is created.

creation-time	description	Date-time of creation of the Container. Note that ideally, this would be the date the Container was first created, which, if the Container was downloaded from a remote location, may not be the same as the local creation date.
	standards	ISO 8601
	validation	As per standard.
	automation	For plain files, this can be obtained from the OS API (stat() on Unix systems).

modification-time	description	Date-time of last modification of the Container.
	standards	ISO 8601
	validation	As per standard.
	automation	The Last-Modified HTTP header contains this information. For plain files, this can be obtained using the OS API (e.g. stat() on Unix systems).

update-time	description	Date-time that information on this container was last updated.
	standards	ISO 8601
	validation	As per standard.
	automation	This can be populated using the GETDATE() command in SQL, or using the API of the host to get the current date and time (e.g. `date +'%Y-%m-%dT%H:%M:%S'`). Since this assumes that the server supplying the date has the correct time, ideally when running the system, the local (UTC) date should be compared with a value returned from an external server (e.g. using the `ntpq` command on Unix).

hash	description	A hash of the contents of the Container.
	standards	None
	validation	Text string of fixed length (e.g. CHAR(256)), consisting of a string

		defining the algorithm used (e.g. SHA-1, MD5), followed by a colon, followed by an encoding string (e.g. hex, base64), followed by a colon, followed by the result of the hash algorithm.
	automation	The MD5 and SHA-1 hash can both be obtained in DOS using the `fciv` command. In Unix, the md5, md5sum, sha1, sha1sum, and openssl commands can variously be used to compute the hash. Content-MD5 can be present in the HTTP header.

Relationships

instance	description	ID in the Container-Type table of the type for this container.
	standards	
	validation	Must be an ID in the Container-Type table
	null	If the container type is not known
	automation	If an application needs a container type and a container is used to supply that need as input and the application runs successfully, then the container type ID *may* be inferred for this container if it is NULL. If an application generates output and this container is known to be that output, then the container-type should be inferred using the product-of relationship.

location-application	description	ID in the Application table of the Application this Container is providing a location for (if any)
	standards	None
	validation	ID in the Application table
	null	If not the location of an Application
	automation	Provided by the user. If the system copies the location provided by the user (e.g. if making a local copy from a remote source), then the ID is inherited from the copied container.

location-documentation	description	ID in the Documentation table of the Documentation this Container is providing a location for (if any)
	standards	None
	validation	ID in the Documentation table
	null	If not the location of Documentation
	automation	None.

generated-by	description	ID in the Study table of a Study this Container has been generated-by (if any)
	standards	PROV:was-generated-by
	validation	ID in the Study table; see also automation here and validation in repository-of.
	null	If not associated with a Study
	automation	The user, when initiating an Application (or Pipeline), may associate the activity with a Study. All files produced by that activity must then have the generated-by field autopopulated with the chosen Study. If the Container is in a collection, and the Container for that collection has a Study via generated-by or repository-of, then the Container *may* inherit that Study. If it already has a Study, the Container's Study must be a part of (transitively) or equal to the Study of the Container collecting it.

repository-of	description	Certain types of Container act as repositories for a Study.
	standards	
	validation	If both generated-by and repository-of are not null, then the Study in the generated-by field must be a part of (transitively) or equal to the Study in the repository-of field.
	null	If this Container is not a repository of any Study.
	automation	None.

held-by	description	ID in the Person table of the person holding this Container. (The owner in the sense that they are the one in whose filesystem, database or webpage the Container appears.)
	standards	PROV:wasAttributedTo
	validation	ID in the Person table
	null	If who holds the Container isn't really relevant or of interest.
	automation	For plain files, the user ID can be used to look up via the account relationship which Person holds the Container.

sourced-from	description	ID in the Person table of the person from whom this Container was sourced, if relevant.
	standards	PROV:wasAttributedTo
	validation	ID in the Person table
	null	If the person from whom the Container was sourced isn't relevant or of interest.
	automation	If a Container is copied, sourced-from should be preserved in the copy. If a Container is copied, and the sourced-from field in the original is NULL, then the sourced-from field in the copy should be set to the held-by field in the original.

output-of	description	ID in the Process table of a process that produced this container as output, if any.
	standards	PROV:wasGeneratedBy
	validation	ID in the Process table
	null	If the Container was not produced by any Process (of interest).
	automation	When an Application (or Pipeline) is run, an entry in the Process table is created, and all output files from that Process will have their Containers' output-of field set to that Process ID. This is essential to recording provenance chains from the use of the system.

collection	description	Simple mereological relationship for Containers. A Container may be part

		of up to one specified other Container – e.g. a file is part of a folder, or a table part of a database. This is a transitive relationship.
	standards	
	validation	ID in the Container table.
	null	If the Container is not part of another Container of interest, or when autopopulating, if it is ambiguous which collecting Container this Container might be part of.
	automation	If the Container is generated-by a Study, and there exists a most-specific Study of which that Study is a part or equal that has repositories, and one of those repositories can be uniquely identified as being a Container for this Container, then the collection should be autopopulated. For example, suppose this Container is a plain file and has been generated by a Study that has a number of folders as repositories. If the file appears in one of the folders, then the collection should be set to the Container ID of that folder.

ContainerType

Description

ContainerTypes are used to collect together Containers of specified types and to infer potential inputs of or outputs from Processes by running Applications. ContainerTypes are also used to state fine-grained metadata through having Variables in their Contents. Not all Containers will have a ContainerType therefore. The Containers that do are those that are associated with Applications.

Standards

None. Note, however, that a ContainerType is a *potential* PROV:Entity in the sense that an Application that has this as one of its Products would, if run in a Process, produce a Container that is an instance of this ContainerType.

Automation

None. When entering an Application in the system, the user is expected to describe all associated ContainerTypes. However, when new Containers are created by running Applications, the system is expected to use the information

here and in associated tables to populate the instance field with the appropriate ContainerType.

Various fields and relationships are used to help with inferring ContainerTypes for Containers. The main means for this is the Product and Uses tables (but especially the Product table). The Product table is a bridge between Application and ContainerType, providing information on how to identify which Container is associated with which output from a Process running an Application. The Uses table operates in a similar way, but for input files to Processes; however, ContainerTypes for input files should only be used where their Containers have no ContainerType identified, if at all. The identifier field provides rules for inferring ContainerType where information cannot be derived using the Product and Uses tables.

Attributes

id_container_type	description	Identifier for the container_type
	standards	None
	validation	Unique within container_types.
	automation	Should be automatically generated by the software framework as the primary key.

format	description	A MIME format text string used to describe the format of the file. An extension may be needed to provide for specific sub-subtypes, such as types of CSV file. The MIME string might not include parameters as the encoding field does in the Container (e.g. charset).
	standards	MIME
	validation	String with maximum length (e.g. CHAR(128)).
	automation	None.

identifier	description	A string providing guidance on how to identify that a file has this type.
	standards	None
	validation	An identifier method string, followed by a colon, followed by a string containing information used by the identifier method to identify the file. Multiple such strings can be separated with a semicolon. For example, numbers of standard binary file formats have the first *n* bits as 'magic numbers' (e.g. for PNG, the first 4 bytes are 0x89504e47).

		The format for the identifier would then be magic:4byte=0x89504e47 (or equivalently magic:32bit=0x89504e47). For CSV files, we may be interested in specific subtypes of CSV file produced by applications. These could be identified both by naming conventions and by the text used for the header row (if there is one). For example, the occupancy file output by FEARLUS-SPOMM, by convention has –lspp.csv as the suffix and first line is "Step,Patch number,Number of Species". This would be represented as the identifier string "magic:line1=Step,Patch number,Number of Species;name:glob=*-lspp.csv"
	automation	None.

Relationships
None.

Content

Description
Content acts as a bridge from ContainerType to Variable, specifying that a Variable appears in some or all instances of a ContainerType. It is associated with fine-grained metadata.

Standards
None

Automation
None. The user is expected to enter this information. There is the possible extension of automatically inferring Content for some ContainerTypes (e.g. CSV files) by assuming the first line contains a comma-separated list of Variables. However, the names of the Variables may not be the same as the headings given them.

Attributes
optionality	description	A statement of whether the Content always or sometimes contains the Variable in the ContainerType.
	standards	None
	validation	The string 'always' if the Variable always appears in the

		ContainerType; 'depends' if the Variable does not always appear in the ContainerType.
	automation	None

locator	description	How to find Values for the Variable in the ContentType, using a formatted string that will be interpreted by tools processing the Container. For now, we just support CSV files and tables in databases.
	standards	None
	validation	Formatted rule e.g.: row:3 (all the values in the third row of a CSV file) column:foo (all the values in the column which on the first row in the CSV file is given the label foo) field:bar (in a database, select bar from <identifier of ContainerType>)
	automation	None

space-locator	description	If the Variable has is-space = true, then how to find Values for the spatial Context of the Variable. Here we should support CSV files and databases providing columns for x, y (and z) co-ordinates as well as IDs of polygons in a GIS file. We should also support ARC Grid ASCII files, and possibly other raster image formats, which perhaps should be done with drivers or other external software.
	standards	None
	validation	Formatted rule e.g.: x=row:3 (all the values in the third row of a CSV file) y=column:foo (all the values in the column which on the first row in the CSV file is given the label foo) polygon=field:bar (in a database, select bar from <identifier of ContainerType>)
	automation	None

time-locator	description	If the Variable has is-time = true, then how to find values for the time dimension.
	standards	None

	validation	Formatted rule e.g.: row:3 (all the values in the third row of a CSV file) column:foo (all the values in the column which on the first row in the CSV file is given the label foo) field:bar (in a database, select bar from <identifier of ContainerType>)
	automation	None

link-locator	description	If the Variable has is-link = true, then how to find values for the link ID.
	standards	None
	validation	Formatted rule e.g.: row:3 (all the values in the third row of a CSV file) column:foo (all the values in the column which on the first row in the CSV file is given the label foo) field:bar (in a database, select bar from <identifier of ContainerType>)
	automation	None

agent-locator	description	If the Variable has is-agent = true, then how to find Values for the spatial Context of the Variable. Here we should support CSV files and databases providing columns for x, y (and z) co-ordinates as well as IDs of polygons in a GIS file. We should also support ARC Grid ASCII files, and possibly other raster image formats, which perhaps should be done with drivers or other external software.
	standards	None
	validation	Formatted rule e.g.: row:3 (all the values in the third row of a CSV file) column:foo (all the values in the column which on the first row in the CSV file is given the label foo) field:bar (in a database, select bar from <identifier of ContainerType>)
	automation	None

Relationships

container-type arse	description	ID in the ContainerType table of the ContainerType this Content appears in.
	standards	None
	validation	ID in ContainerType
	null	Not null
	automation	None

statistical-variable	description	ID in the StatisticalVariable table of the StatisticalVariable this Content refers to.
	standards	None
	validation	ID in StatiscialVariable
	null	Null if VisualisationMethod null
	automation	None

variable	description	ID in the Variable table of the Variable this Content refers to.
	standards	None
	validation	ID in Variable
	null	Null if VisualisationMethod not null
	automation	None

visualisation-method	description	ID in the VisualisationMethod table of the VisualisationMethod this Content refers to.
	standards	None
	validation	ID in VisualisationMethod
	null	Null if Variable not null
	automation	None

A name for (one of) the result(s) of a statistical method.

Context

Description

This table describes the Context associated with a value for a variable – does it apply to a particular time, place, link in a network or agent?

Standards

None

Automation

Depending on how the system handles fine-grained data, it will either generate this on the fly using the Variable and Value tables (the latter of which may also

be generated on an as-needs basis by parsing source data), or be populated automatically by processing output files from processes.

Attributes

id_context	description	Identifier for the context
	standards	None
	validation	Unique within contexts.
	automation	Should be automatically generated by the software framework as the primary key.

value	description	The value of the context (i.e. a particular agent-ID, link-ID, timestamp or polygon-ID)
	standards	None
	validation	String of limited length, which may need to be formatted to allow for specific details (e.g. in co-ordinates, x=30 might be the string to say that this (spatial) context gives the value of 30 to the x co-ordinate)
	automation	This should be populated from the Content table if the Variable the linked Value is for has is-agent, is-space, is-time or is-link true.

Relationships

part	description	Provision for a mereology of Context. Spatial relationships could be computed using RCC (e.g. one region is a proper-part of another); temporal relationships could be computed using Allen's temporal relations; agents might have mereology due to being parts of collectives (I am not aware of a standard implementing this); links could be parts of other links via operators such as property chains in OWL-2 (however, we might also want to provide for links being parts of collectives or networks). For now, this is just a placeholder.
	standards	RCC, Allen's temporal relations; OWL-2
	validation	ID in Context table

	null	If this Context is not part-of another Context.
	automation	Would be computed automatically by a possible future extension to the system – but would need rules to determine mereology.

Contributor

Description
Provides an association between Applications and Documentations and people who contribute to them

Standards
dc:contributor; PROV:was-attributed-to

Automation
None

Attributes

contribution	description	Description of the nature of the contribution (e.g. author, designer, programmer, etc.)
	standards	? (surprised that Dublin Core doesn't have 'author')
	validation	Extensible list of options used by others
	automation	None

alias	description	Name of the contributor as recorded in the contribution (which may be different from that in the Person table).
	standards	
	validation	String (of specified maximum length)
	automation	None

Relationships

contributor	description	Person that is the subject of this contributor relationship
	standards	dc:contributor
	validation	ID in Person table
	null	Not null
	automation	None

documentation	description	Documentation that is the object of this contributor relationship
	standards	

	validation	ID in Documentation table
	null	If Application rather than Documentation is the object. Exactly one of documentation and application must not be null.
	automation	None

application	description	Application that is the object of this contributor relationship
	standards	
	validation	ID in Application table
	null	If Documentation rather than Application is the object. Exactly one of documentation and application must not be null.
	automation	None

Dependency

Description

A dependency is a many-to-many relationship between Applications, and is intended to show how Applications require others documented in the system. For example, all Pipelines have dependencies on the 'first' Application and on the 'next' Pipeline. However, there may be other dependencies due, for example, to documented Applications being called from other Applications when the latter are run.

Standards

None

Automation

Whenever the system starts an Application, it should check for dependencies and ensure they are there. For now, the system is expected to just check 'required' dependencies (see optionality field). For other optionality types, if the dependency is not present, the user will be warned that the requested Application may not run successfully.

Dependencies due to Pipelines should be automatically constructed. Other Dependencies may be more challenging to autopopulate, but not necessarily impossible; such functionality may be deemed an extension for now, but could include such things as:

- Using ldd on Unix systems to look for dynamic libraries used by an executable.
- Checking 'use', 'system' and 'exec' statements in Perl scripts, require() or library() calls in R scripts, or import statements in Python scripts. (More generally, exploiting known commands in programming languages that suggest a dependency.)
- Grepping scripts for the Container entries of the plain files containing other known Applications.

For measures such as this, user interaction may still be required as the results returned may not be 'ontologically interesting' as such.

Attributes

optionality	description	A field indicating the degree to which the Dependency is optional. In some cases, for example, a Dependency may only be a requirement if the Application with the dependency is called with particular command-line options.
	standards	None
	validation	The string "required", a string indicating conditions under which the dependency is required (e.g. "option=-something" if the –something command-line option is given, or "arg:3=*.png" if the dependency is required if argument 3 is a PNG file), or the string "optional" if the dependency is not always required, but the conditions of requirement are not easily expressed.
	automation	None

Relationships

dependant	description	ID of the Application that depends on another Application
	standards	None
	validation	ID in Application table
	null	Not null
	automation	None

dependency	description	ID of the Application that is depended on by another Application
	standards	None (but Google dependee if you want to see some of the debate about what the object of a dependency should be called)
	validation	ID in Application table
	null	Not null
	automation	None

Documentation

Description
Documentation of software or studies. This could be a user guide, some research notes, a diary, a conference or journal paper, or a book.

Standards
Dublin Core metadata standard; PROV:Entity
(Inherited from Dawn's metadata schema.)

Automation
This information must be provided by the user.

Attributes
Note that other attributes from Dublin Core metadata could be applied here.

id_documentation	description	Identifier for the documentation
	standards	None
	validation	Unique within for documentation entries
	automation	Should be automatically generated by the software framework as the primary key.

title	description	Title of the documentation
	standards	dc:title
	validation	String
	automation	None

date	description	Date the documentation was created
	standards	dc:created
	validation	ISO 8601 (string)
	automation	None

Relationships

documents	description	Application this documentation provides documentation for
	standards	
	validation	ID in Application table
	null	If the documentation is not documentation for an Application
	automation	None

references	description	Study this documentation provides documentation for
	standards	
	validation	ID in Study table

	null	If the documentation is not documentation for a Study
	automation	None

Employs

Description

Reified relationship between StatisticalMethods or VisualisationMethods and a StatisticalVariable, recording that the method uses the output of another StatisticalMethod as its input.

Standards

None

Automation

None

Attributes

None

Relationships

statistical-method	description	StatisticalMethod that uses the result of another StatisticalMethod
	standards	None
	validation	ID in StatisticalMethod table
	null	Null if and only if visualisation-method not null
	automation	None

visualisation-method	description	VisualisationMethod that uses the result of a StatisticalMethod
	standards	None
	validation	ID in VisualisationMethod table
	null	Null if and only if statistical-method not null
	automation	None

statistical-variable	description	StatisticalVariable used by the statistical or visualisation method
	standards	None
	validation	ID in Assumption table
	null	Not null
	automation	None

Entailment

Description
Reified relationship between Assumptions and Statistics, noting which Statistics entail which Assumptions.

Standards
None

Automation
None

Attributes
None

Relationships

statistical-method	description	StatisticalMethod that entails an assumption
	standards	None
	validation	ID in StatisticalMethod table
	null	Null if and only if visualisation-method not null
	automation	None

visualisation-method	description	VisualisationMethod that entails an assumption
	standards	None
	validation	ID in VisualisationMethod table
	null	Null if and only if statistical-method not null
	automation	None

assumption	description	Assumption entailed by the statistic
	standards	None
	validation	ID in Assumption table
	null	Not null
	automation	None

Implements

Description
Reified relationship between StatisticalMethods or VisualisationMethods and Applications that implement them.

Standards
None

Automation

None. In future versions, separate tables could be provided giving information about function calls in statistical and visualisation software packages that implement each method. Using this information, applications could be checked for calls to these functions, thereby allowing autopopulation of this table.

Attributes

function	Description	Function in the language of the Application or provided by the application that implements the method
	Standards	None
	Validation	Free text
	Automation	None

library	Description	Library containing the function if appropriate
	Standards	None
	Validation	Free text
	Automation	None

Relationships

statistical-method	Description	StatisticalMethod implemented
	Standards	None
	Validation	ID in StatisticalMethod table
	Null	Null if and only if visualisation-method is not null
	automation	None

visualisation-method	description	VisualisationMethod implemented
	standards	None
	validation	ID in VisualisationMethod table
	Null	Null if and only if statistical-method is not null
	automation	None

application	description	Application implementing the method
	standards	None
	validation	ID in Application table
	Null	Not null
	automation	None

Input

Description
Reified relationship between Process and Container, for capturing provenance about resources used by Processes.

Standards
PROV:used

Automation
For each Dependency of an Application that a Process is running, a specific Container should be identified and recorded as an input with usage = 'dependency'; for each Uses of an Application, a specific instance of the ContainerType should be identified and recorded as an input with usage = 'data'.

Attributes

usage	description	Type of usage, either 'dependency' or 'data'
	standards	None
	validation	One of 'dependency' or 'data'
	automation	Populated when a Process is run.

Relationships

process	description	Process this input pertains to
	standards	None
	validation	ID in Process table
	null	Not null
	automation	Populated when a Process is run.

container	description	Container with the input in it
	standards	None
	validation	ID in Container table
	null	Not null
	automation	Populated when a Process is run.

Involvement

Description
Reified relationship between Persons and Studies, showing what role the Person had in the Study.

Standards
PROV:was-associated-with

Automation
Entered by the user.

Attributes

role	description	The role the person had in the study

	standards	None
	validation	Free text
	automation	None

Relationships

person	description	The subject of the involvement relationship
	standards	None
	validation	ID in Person table
	null	Not null
	automation	None

study	description	The object of the involvement relationship
	standards	None
	validation	ID in Study table
	null	Not null
	automation	None

Meets

Description

Reified relationship between pairs of Specifications

Standards

None

Automation

When a Process is run, the Requirements of its Application (and all (transitive) Dependencies of that Application) should be checked against the Specification of the Computer. The Meets relation therefore only holds between Specifications that apply to Requirements and those that apply to Computers. For each requirement of the Application:

- If it is an 'exact' requirement, then the computer must have a Specification with the same label and value as the requirement.
- If it is a 'match' requirement, then the computer must have a Specification with the same label, the value of which matches the regular expression stored as the value of the requirement Specification.
- If it is a 'minimum' requirement, then the computer must have a Specification with the same label, the parsed numerical value of which is more than or equal to the parsed numerical value of the requirement Specification.

For any requirement for which a corresponding Specification for the computer cannot be found (i.e. there is none with the same label), the system could either terminate (asking the user to supply the information about the computer), or attempt to run the Application, and if it terminates successfully, add an entry in

the Specification table for the Computer accordingly. (The latter would need to be by extension.)

It is possible that this table would only ever have a 'virtual' presence in the database, with all checking done at run time, or when a query is done to see what Applications a Computer is known to be able to run.

Attributes

None

Relationships

computer_specification	description	Specification provided by a Computer that meets the requirement Specification
	standards	None
	validation	ID in Specification table
	null	Not null
	automation	See above.

requirement_specification	description	Specification of a Requirement that is met by the computer Specification
	standards	None
	validation	ID in Specification table
	null	Not null
	automation	See above.

Model

Description

This table is not strictly required by the system. It is more there to indicate that an Application has the 'special status' of being a model rather than a script to set up, run, or process output from a model. This provides a hook for possible future extensions or links to other metadata standards applying to models themselves. Not least among such links would be to models in the CoMSES-Net database.

Standards

PROV:Entity

Automation

None

Attributes

Relationships

id_model	description	Identifier for the model
	standards	None

	validation	Unique within for model entries
	automation	Should be automatically generated by the software framework as the primary key.

application	description	Link to the application implementing the model.
	standards	None
	validation	ID in Application table
	null	Not null
	automation	None

comses	description	Link to the table in the CoMSES-Net database containing information about the model
	standards	None
	validation	ID in appropriate CoMSES-Net table
	null	If no entry in the CoMSES-Net table exists for the model
	automation	None

Parameter

Description

A Parameter is the name of a parameter taken by a statistical or visualisation method, used to configure the way it behaves. For example, in the case of R's rpart() function, parameters are the data stored in the rpart.control() list.

Standards

None

Automation

None

Attributes

id_parameter	description	Identifier for the parameter
	standards	None
	validation	Unique within for parameter entries
	automation	Should be automatically generated by the software framework as the primary key.

data-type	description	Data type of the statistical variable
	standards	Could use XSD
	validation	Text or URI for XSD type if XSD is used
	automation	None

Relationships

statistical-method	description	StatisticalMethod that this is a parameter for
	standards	None
	validation	ID in StatisticalMethod table
	null	Null if and only if visualisation-parameter is not null
	automation	None

visualisation-method	description	VisualisationMethod that this is a parameter for
	standards	None
	validation	ID in VisualisationMethod table
	null	Null if and only if statistical-parameter is not null
	automation	None

Person

Description

A record of the people involved in or associated with the activities and entities recorded by the system. Minimum details are stored about each person, with the PersonalData table being used to record other data.

Standards

PROV:Agent; foaf:Person

Automation

None

Attributes

id_person	description	Identifier for the person
	standards	None
	validation	Unique within for person entries
	automation	Should be automatically generated by the software framework as the primary key.

email	description	Email address of the person
	standards	None
	validation	Free text
	automation	None

Relationships

None

PersonalData

Description
Other personal data of potential interest for recording purposes and studies of people who use the system, but not used by the system.

Standards
None

Automation
None

Attributes

id_personal_data	description	Identifier for the personal data.
	standards	None
	validation	Unique within for personal data entries.
	automation	Should be automatically generated by the software framework as the primary key.

label	description	Label for the information being recorded
	standards	None
	validation	Extensible list of entries used by other users
	automation	None

value	description	Value for the information being recorded
	standards	None
	validation	None
	automation	None

Relationships

about	description	Person this PersonalData is about
	standards	None
	validation	ID in Person table
	null	Not null
	automation	None

Pipeline

Description
A Pipeline is a sequence of Applications to be executed, and constitute a simplified Workflow. Pipelines are essentially implemented using a list-like structure, the head of which is the first Application to run in the Pipeline, and the

tail of which is a reference to a Pipeline for the rest of the Applications to run. It is assumed that each step in the Pipeline must complete before the next step can be started.

Standards

There are a large number of standards for Workflow tools, maintained by different communities.[2] Many of them provide supporting infrastructure, and exceed our requirements, but are also in various levels of development, have various levels of adoption in the community, and it is not yet clear what we should adopt, particularly given that 'Workflow' might be rather too 'grand' a term to use for what we have in mind.

- SCUFL2, for use with Taverna 3, is an Apache incubator project.[3]
- PNML (PetriNet Markup Language).[4] Have a metadata standard that is in the process of standardisation by ISO, but using their *schematron* standard for expressing schemas.
- The OMG have the Business Process Model and Notation[5] that goes somewhat beyond what we need in including humans as part of the workflow.

Whether workflows are appropriate or not for representing aggregations of activities in processing simulation outputs, the idea of this table is to capture specific pathways by which certain kinds of output have been generated to facilitate replication. For now at least, all Pipelines consist exclusively of Applications, which when executed, will be PROV:Activities that are run without human intervention.

Automation

One of the main purposes of the system is to show what others have done to analyse the output of their models, and enable users to repeat the exercise with their own data. The schema described in this document therefore covers both what *has been done* (provenance) and what *could be done*. To some extent, rather than having fixed predefined workflows that people have to adopt, identifying potential activities could be a process involving the exploration of what other users have done in comparable circumstances. Workflows would, under this conceptualisation, 'emerge' from the paths taken by previous researchers. The system could therefore do things like the following:

- Identify the set of Applications that could be run given the ContainerTypes of the currently available data.
- Indicate for each member of that set the number of times each Application has been run by different (groups of) people. (Grouping people would require an extension to the schema.)

---

[2] See http://www.dcc.ac.uk/resources/briefing-papers/standards-watch-papers/workflow-standards-e-science

[3] http://taverna.incubator.apache.org/documentation/scufl2/

[4] http://www.pnml.org/

[5] http://www.omg.org/spec/BPMN/2.0/PDF/

- Indicate comparability of circumstances, perhaps using aggregate variables from fine-grained metadata. (e.g. population of size X, geographical extent of size Y.)

Attributes

id_pipeline	description	Identifier for the pipeline
	standards	None
	validation	Unique within for pipeline entries
	automation	Should be automatically generated by the software framework as the primary key.

Relationships

first	description	The Application to run to start this Pipeline
	standards	None
	validation	ID in Application table
	null	Not null
	automation	None

next	description	The Pipeline to run to do the rest of this Pipeline
	standards	None
	validation	ID in Pipeline table
	null	If there are no further Applications to run in the Pipeline
	automation	None

Process

Description

Processes are a record of a specific run of an Application. All instances of this table are generated automatically by the system when an Application is run. This is crucial to recording the provenance of how results have been analysed.

Standards

PROV:Activity

Automation

When the user requests an Application to be run on a Computer, the system does the following:

1. Checks that a Container can be found for it.
2. Checks that the Computer meets the Requirements of the Application.
3. Checks that Containers can be found for all (transitive) Dependencies.
4. Checks that the Computer meets the Requirements of all Dependencies.

5. Checks that command-line arguments have been specified. (See Automation under Arguments.)
6. Checks that Container instances are present for all ContainerTypes the Application Uses (if necessary, prompting the user).
7. Builds the call to the shell.
8. Gets the user and host.
9. Notes the date/time.
10. Makes the call to the shell.
11. Notes the date/time after the call has terminated.

Attributes

id_process	description	Identifier for the process.
	standards	None
	validation	Unique within for process entries.
	automation	Should be automatically generated by the software framework as the primary key.

start-time	description	Date-time the process was started, with fractions of a second if that information is available.
	standards	ISO 8601
	validation	As per standard
	automation	On some platforms, the %N format in the `date` command gives nanoseconds.

end-time	description	Date-time the process ended, with fractions of a second if that information is available.
	standards	ISO 8601
	validation	As per standard
	automation	On some platforms, the %N format in the `date` command gives nanoseconds

argv	description	Record of the string passed to the shell to run the command
	standards	None
	validation	String
	automation	None

environment	description	Record of the setting of relevant environment variables when running the command
	standards	None
	validation	List of string

	automation	For each element of the envs list in the Application, record the value of the environment variable as VAR=value. On Unix systems, specific environment variables can be accessed using `echo` or all environment variables can be accessed using `env`.

working-directory	description	Record of the working directory of the user at the time the Process was run
	standards	None
	validation	String
	automation	Can be accessed on Unix systems with `pwd`.

Relationships

executable	description	Application this Process is running
	standards	PROV:used
	validation	ID in the Application table
	null	Not null
	automation	Recorded when the Process is initiated.

user	description	User this Process is being run for
	standards	PROV:wasAssociatedWith
	validation	ID in the User table
	null	Not null
	automation	Recorded with the Process is initiated. OS and programming language APIs typically provide access to this information. e.g. on Unix systems via the getuid() function.

host	description	Computer this Process is being run on
	standards	PROV:wasAssociatedWith
	validation	ID in the Computer table
	null	Not null
	automation	Recorded when the Process is initiated.

parent	description	(For possible future extensions.) If an Application calls another Application, details of that could be recorded provided appropriate

		information can be accessed about that call from within the system.
	standards	None
	validation	ID in the Process table
	null	If the Process has no parent that is recorded in the Process table.
	automation	Would need to be achieved using system API that enabled watching processes initiated here to see if they spawned any children.

Product

Description

A Product is a reified relationship between ContainerTypes and Applications, indicating that an Application produces Containers of a particular ContainerType as output.

Standards

None

Automation

None

Attributes

optionality	description	A string indicating whether or not the Application always produces a Container of this ContainerType as output.
	standards	None
	validation	'always' if the Application always produces a Container of this ContainerType; 'depends' otherwise.
	automation	None

locator	description	A string indicating how to find this ContainerType among the outputs produced by the Application.
	standards	None
	validation	See Uses.locator for more information.
	automation	None

Relationships

application	description	Application making this Product
	standards	None
	validation	ID in Application table

	null	Not null
	automation	None

container-type	description	ContainerType of this Product
	standards	None
	validation	ID in ContainerType table
	null	Not null
	automation	None

in-file	description	If the locator is 'in-file', then this contains an ID in the ContainerType table of a ContainerType in which to find the location of the Product.
	standards	None
	validation	The ID of the ContainerType MUST appear in an existing entry in the Uses or Product table having the same application ID entry.
	null	If the locator is not 'in-file'
	automation	None

Project

Description
The project associated with some output. A Project is a special Study that has funding and grant-ID associated with it.

Standards
PROV:Activity
Inherited from Dawn's metadata table

Automation
User entry

Attributes

id_project	description	Identifier for the project.
	standards	None
	validation	Unique within for project entries.
	automation	Should be automatically generated by the software framework as the primary key.

title	description	A title for the project
	standards	None
	validation	String
	automation	None

funder	description	Name of a funding body for this project
	standards	None
	validation	String
	automation	None

grant-ID	description	Grant number for the project, or other identifier used by the funding body to refer to it
	standards	None
	validation	String
	automation	None

Relationships
None

Requirement

Description
A Requirement is a Specification needed by an Application that a Computer must meet exactly or minimally, or match in order to run it. (e.g. an amount of RAM.)

Standards
None, though I wonder if some of the workflow or web service architecture standards might have a model we could draw on.

Automation
Entered by the user. The absence of a Requirement should not be taken to infer that there is none; rather that it is either not known or that it has not been entered for some other reason.

Attributes
None

Relationships

application	description	Application this Requirement is for
	standards	None
	validation	ID in the Application table
	null	Not null
	automation	None

match	description	Specification that a Computer must match in order to meet the Requirement. The value of the Specification ID is a POSIX conforming regular expression.
	standards	None
	validation	ID in the Specification table

	null	If the Requirement is instead a minimum or exact requirement
	automation	None

minimum	description	Numerical Specification that a Computer must have at least that value for in order to meet the Requirement. The value must be parseable as a number
	standards	None
	validation	ID in the Specification table
	null	If the Requirement is instead a match or exact requirement
	automation	None

exact	description	Specification that a Computer must have the exact value of in order to meet the Requirement. The value of the Specification ID is a string.
	standards	None
	validation	ID in the Specification table
	null	If the Requirement is a match or minimum requirement
	automation	None

Specification

Description
A label and value for a Specification that is the subject either of a Requirement of an Application or of a Computer.

Standards
None

Automation
None

Attributes

id_specification	description	Identifier for the specification.
	standards	None
	validation	Unique within for specification entries.
	automation	Should be automatically generated by the software framework as the primary key.

label	description	String label describing what the value of the Specification is. This should be an extensible list allowing previous entries to be re-entered.
	standards	None
	validation	String
	automation	None

value	description	Value for the label. If of a match Requirement, the value should be a regular expression string; if of a minimum requirement, the value should be parseable as a number.
	standards	POSIX ERE standard (ISO/IEC/IEEE 9945:2009) for regular expressions if this is the Specification of a match Requirement.
	validation	String, possibly a regular expression or a parseable number – see above.
	automation	None

Relationships

specification-of	description	Computer this is a Specification for, if this is a Specification for a Computer rather than a Requirement
	standards	None
	validation	ID in Specification table
	null	If this is the Specification of a Requirement
	automation	None

StatisticalInput

Description

Reified relationship between Statistics or Visualisation and Value, recording when the result-of a Statistics is used by another Statistics or a Visualisation. If the StatisticalMethod or VisualisationMethod used by the Statistics or Visualisation Employs a StatisticalVariable generated-by a StatisticalMethod, then there should be an entry in this table recording the actual result used.

Standards

PROV:Used

Automation

A StatisticalInput entry should be created each time a Statistics or Visualisation has a corresponding StatisticalMethod or VisualisationMethod that Employs a StatisticalVariable. In some cases, the Value used may not be present (because the appropriate StatisticalMethod has not already been applied). In such cases, the system would ideally automatically generate the Value, and it may be known that the Value should simply be computed from the same source data as that required for the application of this StatisticalMethod or VisualisationMethod.

Attributes

None

Relationships

statistics	description	Statistics using a Value that is the result of another Statistics
	standards	None
	validation	ID in Statistics table
	null	Null if and only if visualisation not null
	automation	None

visualisation	description	Visualisation using a Value that is the result of a Statistics
	standards	None
	validation	ID in Visualisation table
	null	Null if and only if statistics not null
	automation	None

value	description	Value being used
	standards	None
	validation	ID in Value table, the entry for which must have result-of not null
	null	Not null
	automation	None

StatisticalMethod

Description

A statistical method is an approach to computing some statistics. It may be implemented in or as part of an application. A statistical method generates one or more statistical variables as its results, and may use the results of another statistical method in its computation. For example, computing the standard deviation of some data uses the mean of those data.

Each time a statistical method is applied, a Statistics entry should be created. For each StatisticalVariable the StatisticalMethod Employs, there should be a StatisticalInput entry, and for each StatisticalVariable that is generated-by the

StatisticalMethod, there should be a Value entry with the result-of field containing the ID of the Statistics activity.

Standards
None

Automation
None

Attributes

id_statistical_method	Description	Identifier for the statistical method.
	Standards	None
	Validation	Unique within for statistical method entries.
	automation	Should be automatically generated by the software framework as the primary key.

Relationships
None

StatisticalVariable

Description
A name for (one of) the result(s) of a statistical method.

Standards
None

Automation
None

Attributes

id_statistical_variable	description	Identifier for the statistical variable.
	Standards	None
	Validation	Unique within for statistical variable entries.
	automation	Should be automatically generated by the software framework as the primary key.

data-type	description	Data type of the statistical variable
	standards	Could use XSD
	validation	Text or URI for XSD type if XSD is used
	automation	None

Relationships

statistic-generated-by	description	StatisticalMethod that generates this statistical variable
	standards	None
	validation	ID in StatisticalMethod table
	null	Not null
	automation	None

visualisation-generated-by	description	VisualisationMethod that generates this statistical variable
	standards	None
	validation	ID in VisualisationMethod table
	null	Not null
	automation	None

Statistics

Description

Statistics are activities that compute the Values of StatisticalVariables. They operate on raw data that are retrieved from the Values table using a query. To replicate a Statistics, the query can be rerun, selecting Values that are contained-in Containers the Processes generating which have a start-time earlier than the date the Statistics was done. Since Value is a virtual table, the query might not be in SQL, but could be a shell expression (e.g. using awk, grep or sed) to extract the data to use from a file (Container).

Standards

PROV:Activity

Automation

A record of the Statistics can be automated if it is known that the Statistics used a StatisticalMethod that has an Application that Implements it, and a Process running that Application produced a Container containing Values of StatisticalVariables generated-by the StatisticalMethod.

Similarly, an application wrapping the command-line interface to a statistical package would presumably be capable of populating the Statistics table each time the user ran a command known to implement a StatisticalMethod.

Another approach would be to initiate a Statistics through the user indicating they wanted to apply a StatisticalMethod to some data. The system would allow the user to select the data they wanted to apply the method to, recording the query. It would check whether the StatisticalMethod had any parameters, and ask the user what values they wanted to use for those parameters (recording them in the Value table). It would also check whether the StatisticalMethod used any StatisticalVariables, and then either (a) ask the user which Values of those StatisticalVariables they wanted to use, or (more likely) (b) run an implementation of the StatisticalMethod generating the StatisticalVariables on the source data the user originally selected. In either case the Values of the

StatisticalVariables Employed by the StatisticalMethod the user wants to apply would be recorded as entries in the StatisticalInput table. Finally, for each StatisticalVariable generated by the StatisticalMethod, the results of the Statistics would be recorded as entries in the Value table.

Attributes

id_statistics	description	Identifier for the statistic.
	standards	None
	validation	Unique within for statistics entries.
	automation	Should be automatically generated by the software framework as the primary key.

date	description	Date-time the Statistics was computed.
	standards	ISO 8601
	validation	As per standard
	automation	Timestamp using appropriate shell or API call

query	description	Details of how to retrieve the same raw data used to compute the Statistics
	standards	None
	validation	Formatted string. We do not propose to impose a comprehensive standard for this string to allow extensibility. However, the string should comprise a method of extraction/retrieval, followed by a colon, followed by a command in that method to obtain the data. For example: sql-where:contained-in = *container-ID* shell:awk –F, '{print $2}' *filename* Note that in the SQL case, the first part of the SQL statement should always be SELECT value from Value WHERE …
	automation	Can be autogenerated if appropriately wrapped.

Relationships

used	description	StatisticalMethod this Statistics used

	standards	None (note that this isn't the same sense as PROV:used, as StatisticalMethods are not PROV:Entities)
	validation	ID in StatisticalMethod table
	null	Not null
	automation	None

Study

Description

A Study is a piece of work at some level of aggregation, which allows simulation outputs to be grouped together. Studies can be parts of other Studies. For example, a Study might be a simulation experiment, that is part of another Study to prepare a publication. Multiple studies make up a Project.

Standards

PROV:Activity

Automation

None

Attributes

id_study	description	Identifier for the study.
	standards	None
	validation	Unique within all studies
	automation	Should be automatically generated by the software framework as the primary key.

label	description	A label for the Study
	standards	None
	validation	String
	automation	None

start-time	description	Date-time the Study started
	standards	ISO 8601
	validation	String stored as per standard
	automation	None

end-time	description	Date-time the Study ended
	standards	ISO 8601
	validation	String stored as per standard
	automation	None

Relationships

project	description	If this Study is a Project, then the Project that this Study is

	standards	None
	validation	ID in Project table
	null	If the Study is not a Project
	automation	None

part	description	Study this Study is a (proper) part of, if any
	standards	None
	validation	ID in Study table
	null	If the Study is not a proper part of a Project
	automation	None

Tag

Description
Allow certain things to be tagged.

Standards
Inherited from Dawn's metadata standard

Automation
None

Attributes

id_tag	description	Identifier for the tag.
	standards	None
	validation	Unique within all tag entries.
	automation	Should be automatically generated by the software framework as the primary key.

Relationships
None, but see TagMap.

TagMap

Description
Reified many-many relationship between Tags and the objects tags can be applied to.

Standards
None

Automation
None

Attributes
None

Relationships

At least one of other_tag, container-type, application, container, documentation and study must be not null.

tag	description	Tag this is mapping from
	standards	None
	validation	ID in Tag table
	null	Not null
	automation	None

other_tag	description	Tag this is mapping to
	standards	None
	validation	ID in Tag table
	null	Not null
	automation	None

container-type	description	ContainerType being tagged
	standards	None
	validation	ID in ContainerType
	null	If not a tag of a ContainerType
	automation	None

application	description	Application being tagged
	standards	None
	validation	ID in Application table
	null	If not a tag of an Application
	automation	None

container	description	Container being tagged
	standards	None
	validation	ID in Container table
	null	If not a tag of a Container
	automation	None

documentation	description	Documentation being tagged
	standards	None
	validation	ID in Documentation table
	null	If not a tag of a Documentation
	automation	None

study	description	Study being tagged
	standards	None
	validation	ID in Study table
	null	If not a tag of a Study
	automation	None

	description	StatisticalMethod being tagged

statistical-method	standards	None
	validation	ID in StatisticalMethod table
	null	If not a tag of a StatisticalMethod
	automation	None

visualisation-method	description	VisualisationMethod being tagged
	standards	None
	validation	ID in VisualisationMethod table
	null	If not a tag of a VisualisationMethod
	automation	None

User

Description
A user is an account on a Computer that is owned by a Person.

Standards
PROV:Agent

Automation
None.

Attributes

id_user	description	Identifier for the user.
	standards	None
	validation	Unique within all user entries.
	automation	Should be automatically generated by the software framework as the primary key.

uid	description	Identifier of the User from the perspective of the operating system on the host computer
	standards	None
	validation	String
	automation	Can be accessed using `id` on Unix systems.

home-directory	description	Home directory of the User
	standards	None
	validation	String
	automation	Can be accessed using `echo $HOME` on Unix systems.

Relationships

account-of	description	Person this User is associated with

	standards	PROV:actedOnBehalfOf
	validation	ID in Person table
	null	If the Person is not known
	automation	Difficult to assess automatically. On some Unix systems, there is the convention of using unused fields in the /etc/passwd file to give User's full name, but this is less common now due to security concerns. On Macs, this information is in the pw_gecos element of struct passwd (see `man getpwent`), but even then, knowing the name associated with the account may not be sufficient to identify a Person (e.g. because the Person has a common name, or because the Person is known by a number of different names).

Uses

Description
Reified relationship between Applications and ContainerTypes indicating the kinds of Container that are required by the Application as input

Standards
None

Automation
None

Attributes

optionality	description	Degree to which the ContainerType is required.
	standards	None
	validation	String, one-of: 'required': the content type must be provided to the application for all invocations 'optional': the content type can be used for some invocations 'depends': the content type may be required depending on other inputs to the application.
	automation	None

locator	description	Information on how to specify the ContainerType when building the input to the Application before

		running it. See also the Argument class.
	standards	None
	validation	String, formatted as follows: arg=*X* the container type appears as command-line argument number *X* of the application argID=X the container type appears as command-line ArgumentValue with ID X in the Argument table. opt=*Y* the container type appears as the argument to command-line option with name *Y* of the application env=*Z* the container type appears as the value of environment variable *Z* checked by the application in-file: use in-file relationship – the container type is referred to in another file used as input to the application
	automation	None

Relationships

application	description	Application that Uses the ContainerType
	standards	None
	validation	ID in Application table
	null	Not null
	automation	None

container-type	description	ContainerType that is Used
	standards	None
	validation	ID in ContainerType table
	null	Not null
	automation	None

in-file	description	If the locator is 'in-file', then the ContainerType in which the location of this ContainerType is specified. That entry MUST then have a Uses entry for the same Application.
	standards	None
	validation	ID in the ContainerType table
	null	If the locator is not 'in-file'
	automation	None

Value

Description

Value of a Variable, recorded in some Container. As currently, planned, it is not proposed to store these values in a table; rather, to use the original output data. Hence this table has a 'virtual' presence and will need to be generated as required given a query using it. The format and units attributes are stored as information in the Content of the ContainerType for the Variable the Value is for.

Standards

PROV:Entity

Automation

When a Value is requested, the system will need to look in the appropriate Container to get it.

Attributes

id_value	description	The value of the variable
	standards	None
	validation	String
	automation	None

Relationships

variable	description	Variable this Value is of
	standards	None
	validation	ID in Variable table
	null	Null if and only if either of parameter and statistical-variable are not null
	automation	The Variable this is a Value for should be deducible from the ContainerType fhormat of the Container this Value appears in.

statistical-variable	description	StatisticalVariable this Value is of
	standards	None
	validation	ID in StatisticalVariable table; result-of must be completed if this is not null
	null	Null if and only if either of parameter and variable are not null
	automation	This would be populated as part of creating the Statistics or Visualisation entry.

parameter	description	Parameter this Value is of
	standards	None

	validation	ID in Parameter table; statistical-parameter or visualisation-parameter must be completed if this is not null
	null	Null if and only if either of variable and statistical-variable are not null
	automation	This would be populated as part of creating the Statistics or Visualisation entry.

statistical-parameter	description	Statistics this Value is a parameter value of
	standards	None
	validation	ID in Statistics table; the Statistics entry must use a StatisticalMethod with a statistical-parameter that has the same ID in the Parameter table as the parameter entry in this table.
	null	Not null if and only if parameter not null
	automation	This would be populated as part of creating the Statistics entry.

visualisation-parameter	description	Visualisation this Value is a parameter value of
	standards	None
	validation	ID in Visualisation table; the Visualisation entry must use a VisualisationMethod with a visualisation-parameter that has the same ID in the Parameter table as the parameter entry in this table.
	null	Not null if and only if parameter not null
	automation	This would be populated as part of creating the Visualisation entry.

result-of	description	Statistics this Value is (one of) the result(s) of
	standards	None
	validation	ID in Statistics table; the Statistics entry must use a StatisticalMethod having a StatisticalVariable that is generated-by it. The entry in the statistical-variable field in this table must be one of those StatisticalVariable(s).

	null	Not null if and only if statistical-variable not null
	automation	This would be populated as part of creating the Statistics entry, once the statistics have been computed.

time	description	Temporal Context of the Value, if relevant (as determined by variable having is-time true).
	standards	None
	validation	ID in Context table
	null	If no relevant temporal Context
	automation	If is-time is true, then use time-locator of Content for the ContainerType to find or populate the Context table with the temporal Context of this Value.

space	description	Spatial Context of the Value, if relevant (as determined by variable having is-space true).
	standards	None
	validation	ID in Context table
	null	If no relevant spatial Context
	automation	If is-space is true, then use space-locator of Content.

agent	description	Agent Context of the Value, if relevant (as determined by variable having is-agent true).
	standards	None
	validation	ID in Context table
	null	If no relevant agent Context
	automation	If is-agent is true, then use agent-locator of Content.

link	description	Link Context of the Value, if relevant (as determined by variable having is-link true).
	standards	None
	validation	ID in Context table
	null	If no relevant link Context
	automation	If is-link is true, then use link-locator of Content.

contained-in	description	Container this Value appears in
	standards	None

	validation	ID in Container table
	null	Not null
	automation	Since the Values table is virtual, this would be obvious from the file being read to obtain the value.

Variable

Description

A Variable of interest (or potential interest), Values of which are stored in Containers of certain ContainerTypes.

Standards

None

Automation

None

Attributes

name	description	A name for the variable
	standards	None
	validation	String
	automation	None

data-type	description	A data-type for the variable.
	standards	Could use XSD.
	validation	Text or URI for XSD type if XSD is used.
	automation	None

is-agent	description	If the variable applies to individual agents, then true
	standards	None
	validation	Boolean
	automation	None

is-link	description	If the variable applies to individual links, then true
	standards	None
	validation	Boolean
	automation	None

is-space	description	If the variable applies to specific (artificial or geographical) areas, then true
	standards	None
	validation	Boolean

	automation	None

is-time	description	If the variable applies to a specific (artificial or real) time, then true
	standards	None
	validation	Boolean
	automation	None

Relationships
None.

Visualisation

Description
A Visualisation is the process of creating an image to depict one or more (typically more than one) Values. The results of a visualisation appear in a Container.

Standards
PROV:Activity

Automation
An entry would be created here when the user wants to run a VisualisationMethod, or when an Application that Implements a VisualisationMethod is run as a Process.

Attributes
id_visualisation	description	Identifier for the visualisation.
	standards	None
	validation	Unique within all visualisation entries.
	automation	Should be automatically generated by the software framework as the primary key.

date	description	Date-time the Visualisation was created.
	standards	ISO 8601
	validation	As per standard
	automation	Timestamp using appropriate shell or API call

query	description	Details of how to retrieve the same raw data used to create the Visualisation
	standards	None
	validation	Formatted string. See Statistics.

	automation	Can be autogenerated if appropriately wrapped.

Relationships

visualisation-method	description	The visualisation method used to generate this visualisation
	standards	None
	validation	ID in VisualisationMethod table
	null	Not null
	automation	Known from the Content or from the VisualisationMethod used to initiate the Visualisation activity.

contained-in	description	The Container this Visualisation appears in; the result of this Visualisation.
	standards	None
	validation	ID in the Container table
	null	Not null
	automation	Known from the Product of the Application creating the Visualisation and the corresponding identifier in the ContainerType when the Process is run.

VisualisationMethod

Description

This table describes methods for generating Visualisations, which then may appear in the Content of a Container produced by a Process running an Application that Implements it.

Standards

None

Automation

None

Attributes

label	description	Brief name for the method (the description would contain longer text)
	standards	None
	validation	None
	automation	None

Relationships

None

Index