Work Package 2 : Workflow

Capturing Pharmacometrics Workflow Concepts

Related task of the project (Task # and full name):	
Author:	Jonathan Chard
Prepared by:	
Approved by:	

1. Definitions

1.1 Acronyms and Abbreviations

Acronym	Definition

Table of contents

2 Introduction

2.1 Purpose

2.2 Audience

2.3 Scope

3 PROV-O Terms

PROV-O defines the following terms that are used to capture all the information necessary to define the provenance of items within a document.

Entity

An entity is a physical, digital, conceptual, or other kind of thing with some fixed aspects; entities may be real or imaginary.

It has the following “subtypes”:

- Collection
  - A grouping of Entities, and only entities.
- Plan
  - A plan is an entity that represents a set of actions or steps intended by one or more agents to achieve some goals
- Bundle
  - A bundle is a set of proActivities and Agentsvenance descriptions, so it can contain Entities,
  - Note however that a bundle cannot contain more bundles

Activity

An activity is something that occurs over a period of time and acts upon or with entities; it may include consuming, processing, transforming, modifying, relocating, using, or generating entities.

An **activity**, has:

- *id*: an identifier for an activity;
- *startTime*: an OPTIONAL time (st) for the start of the activity;

- *endTime*: an OPTIONAL time (et) for the end of the activity;
- *attributes*: an OPTIONAL set of attribute-value pairs representing additional information about this activity.

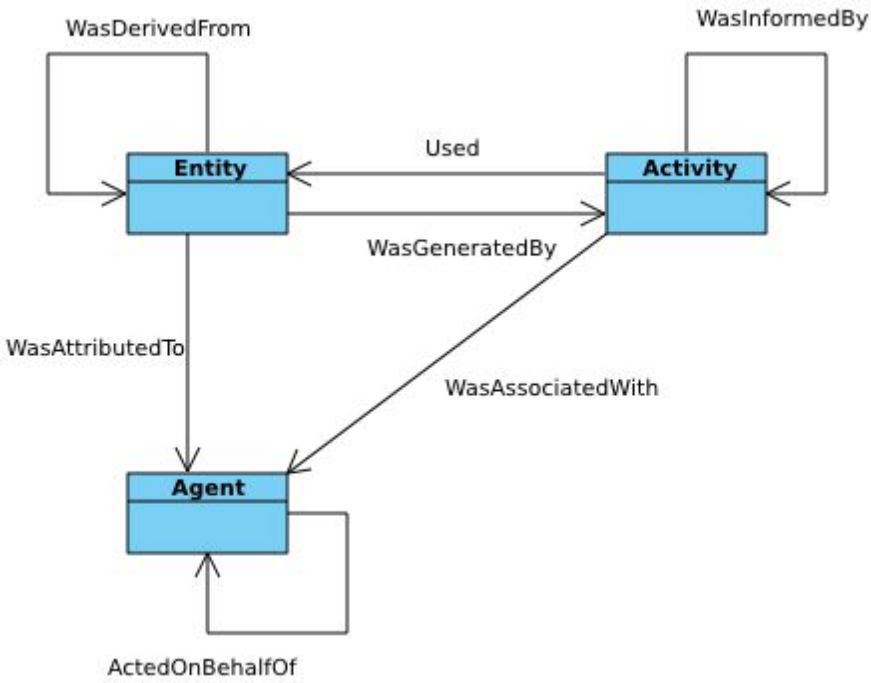
Agent

An agent is something that bears some form of responsibility for an activity taking place, for the existence of an entity, or for another agent's activity.
Subtypes:

- Organisation (e.g. Leiden, Pfizer)
- Person (e.g. a User)
- Software Agent (e.g. R, Monolix)

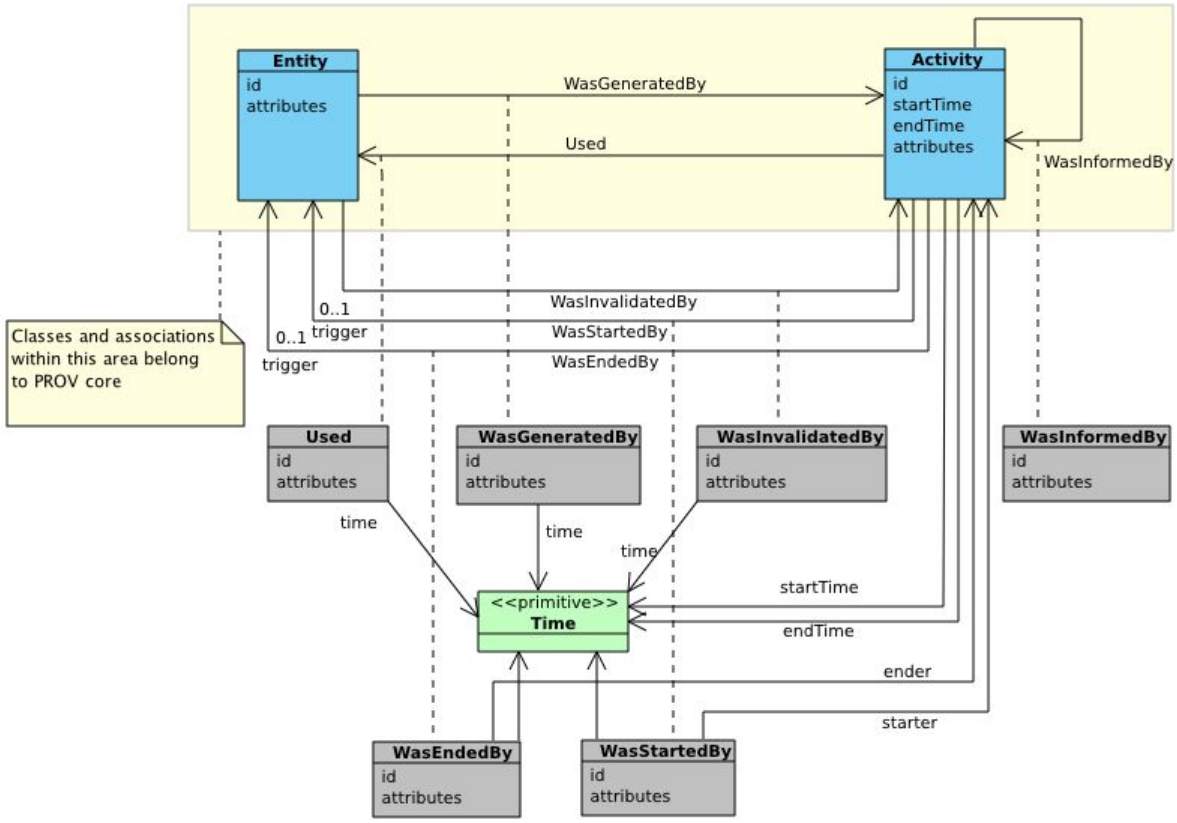
Relationships

PROV-O defines a set of relationships that describe the interactions between each of the entity types above.



Activity - Entity relationships

The following diagram lists the possible relationships that can exist between an activity and an entity.



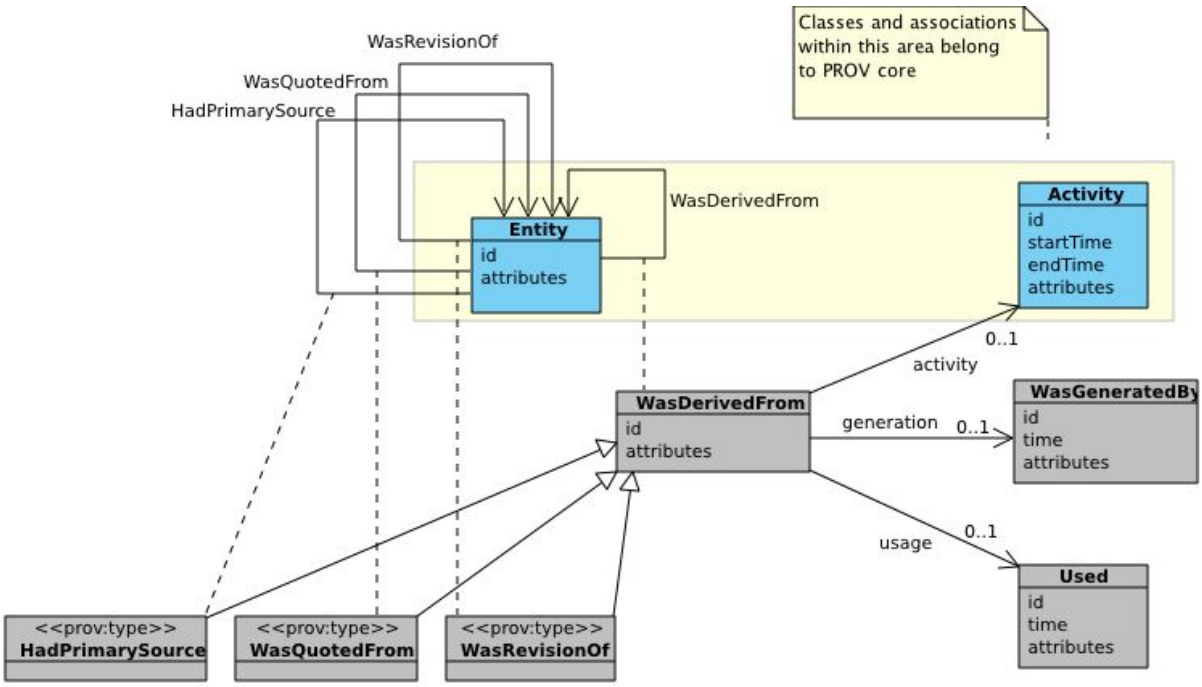
The most relevant to us are:

Was generated by	Generation is the completion of production of a new entity by an activity. This entity did not exist before generation and becomes available for usage after this generation.

Used	Usage is the beginning of utilizing an entity by an activity. Before usage, the activity had not begun to utilize this entity and could not have been affected by the entity
Was invalidated by	Invalidation is the start of the destruction, cessation, or expiry of an existing entity by an activity. The entity is no longer available for use (or further invalidation) after invalidation.
Was started by	**Start** ◊ is when an activity is deemed to have been started by an entity, known as **trigger** ◊. The activity did not exist before its start. Any usage, generation, or invalidation involving an activity follows the activity's start. A start may refer to a trigger entity that set off the activity, or to an activity, known as **starter** ◊, that generated the trigger.
Was ended by	**End** ◊ is when an activity is deemed to have been ended by an entity, known as **trigger** ◊. The activity no longer exists after its end. Any usage, generation, or invalidation involving an activity precedes the activity's end. An end may refer to a trigger entity that terminated the activity, or to an activity, known as **ender** ◊ that generated the trigger.

Entity - Entity relationships

The following diagram indicates in more detail the relationships that can exist between two entities:



There is one type of relationship that can exist between entities (“derived from”), which has three subtypes that we can use:

Term	Strict definition
Derived from	A derivation is a transformation of an entity into another, an update of an entity resulting in a new one, or the construction of a new entity based on a pre-existing entity.
Revision	A revision is a derivation for which the resulting entity is a revised version of some original. The implication here is

	that the resulting entity contains substantial content from the original. Revision is a particular case of derivation.
Quotation	A quotation is the repeat of (some or all of) an entity, such as text or image, by someone who may or may not be its original author. Quotation is a particular case of derivation.
Primary Source	A primary source for a topic refers to something produced by some agent with direct experience and knowledge about the topic, at the time of the topic's study, without benefit from hindsight. Because of the directness of primary sources, they 'speak for themselves' in ways that cannot be captured through the filter of secondary sources. As such, it is important for secondary sources to reference those primary sources from which they were derived, so that their reliability can be investigated. A primary source relation is a particular case of derivation of secondary materials from their primary sources. It is recognized that the determination of primary sources can be up to interpretation, and should be done according to conventions accepted within the application's domain

Derivations are specified as below:

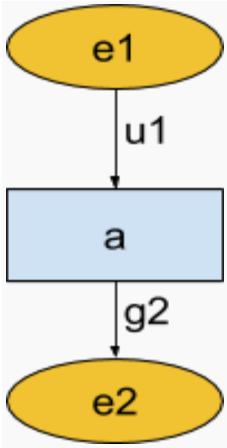
wasDerivedFrom(ex:d; e2, e1, a, g2, u1, [ex:comment="a righteous derivation"])

Here:

- d is the optional derivation identifier, e2 is the identifier for the entity being derived,
- e1 is the identifier of the entity from which e2 is derived,
- a is the optional identifier of the activity which used/generated the entities,

- g2 is the optional identifier of the generation,
- u1 is the optional identifier of the usage, and
- [ex:comment="a righteous derivation"] is a list of optional attributes. In PROV-N these fields are used to capture the type of derivation, i.e.:
  - revision - `prov:type='prov:Revision'`
  - quotation - `prov:type='prov:Quotation'`
  - primary source - `prov:type='prov:PrimarySource'`

Hence we can capture the activity that generated the derivation if we wish.



Activity - Activity relationships

It is also possible to have relationships between activities.

These are summarized below:

Was informed by	Some anonymous entity passes between two activities.

	So, the prov:wasInformedBy property allows the construction of provenance chains comprising only Activities
Acted on behalf of	Delegation is the assignment of authority and responsibility to an agent (by itself or by another agent) to carry out a specific activity as a delegate or representative, while the agent it acts on behalf of retains some responsibility for the outcome of the delegated work.

Entities - agent relationships

The following table lists the types of relationship that can exist between an agent and an entity

Attributed to	Attribution is the ascribing of an entity to an agent
Associated with	An activity association is an assignment of responsibility to an agent for an activity, indicating that the agent had a role in the activity. It further allows for a plan to be specified, which is the plan intended by the agent to achieve some goals in the context of this activity.

Object - object relationships

All the above relationships are subtypes of one relationship - “wasInfluencedBy”. This can be used to ascribe any relationship between any objects. The W3C recommends not to use this relationship, but we can use it if necessary - though we should attach appropriate attributes to the relationship to specify it properly.

4 Pharmacometrics Workflow Concepts

Entities

According to the PROV-O specification, an entity is “a physical, digital, conceptual, or other kind of thing with some fixed aspects; entities may be real or imaginary.”

When an “entity” is defined, it normally does not represent the entire entity itself, but indicates where it can be located or retrieved later, i.e. it is a representation of that entity.

This definition works perfectly on entities that can be directly mapped to files within a pharmacometric project, for instance:

- a dataset
- a NONMEM control file
- an R script
- an output file (e.g. generated by NONMEM or Monolix)
- a graphical output (e.g. a PNG)
- a report (e.g. a Word document or a PDF)

These types of entity can always be retrieved from a secure storage location (e.g. a version control system) at any point in the future, subject to proper backup and recovery procedures, as long as the ID of the entity can be traced back to the individual file (and version of that file).

This definition works less well on “imaginary” or “conceptual” entities as they do not “exist” (as much as anything digital exists), and cannot easily be retrieved at a later date.

Examples of “imaginary” or “conceptual” entities would be:

- a decision
- an assumption
- a data/model/parameter/task object defined within an MDL file

Capturing this type of entity will require creation of a file that represents that entity and contains information pertinent to it, which is added to the version control system.

The provenance ontology standard requires a minimal amount of information to define entities - all that is necessary is an id, which can be used to universally, uniquely identify that entity. It is also possible to attach any arbitrary attributes to an entity to describe it within the system.

The entity id can also be prefixed with a namespace that can be combined with the ID to further identify it.

The ontology also predefines a number of attributes that may be used to define an entity:

- prov:label (0 or more)
- prov:location (0 or more)
- prov:type (0 or more)
- prov:value (0 or 1)

Usage of existing attributes

Namespace

We will use the “namespace” to define the URL of the repository in which the entity can be found.

We will use the “repo” as the shorthand name of the namespace within each provenance document.

Each provenance document will include the following namespaces:

default <http://ddmore.eu/workflow/>
prefix ddmore <http://ddmore.eu/workflow/>
prefix repo <https://github.com/johndoe/examplerepo.git/>

Provenance Attributes

The following table summarizes the types of entity we will track in the system. These will be captured using the “prov:type” attribute.

Entity type	**Examples**	**Metadata to describe it**
Model	NONMEM control file (.ctl, .mod) MDL file (.mdl) Monolix model (.mlxtran)	prov:type=ddmore:model
Dataset	CSV (.csv) Table file (.tab) Data file (.dat)	prov:type = ddmore:dataset
PharmML archive	.phex	prov:type = ddmore:phex
Standard output object	.so	prov:type = ddmore:so
Output	NONMEM output file (.lst, .out)	prov:type =

	Monolix output file Output tables	ddmore:output
Image	PDF image PNG image JPG image (etc)	prov:type = ddmore:image
Assumption	XML document	prov:type = ddmore:assumption
Decision	XML document	prov:type = ddmore:decision
Document	HTML document DOCX document PDF document RTF document (etc)	prov:type = ddmore:document

“prov:location” will be used to identify the location within the version control system of the file that this entity represents.

“prov:label” is designed to “provide a human-readable representation of an instance of a PROV-DM type or relation”. A number of labels can be attached.

Therefore we will use “prov:label” to provide a human readable description of the entity or activity.

There are a number of other states/flags that we wish to be able to attach to entities, to identify:

- Importance/Status of an output
- QC status

- "Significance" within the project

We will define our own properties to do this, so that they can be namespaced to ddmore.

These are:

Metadata field name	Meaning	Possible values
ddmore:qcStatus	Whether this entities has passed or failed a QC process	<blank> (indicates it has not been QCed) true (passed QC) false (failed QC)
ddmore:final	This is the final model	true / false
ddmore:base	This is the base model	true / false
ddmore:pivotal	This model is pivotal	true / false
ddmore:outputType		primary secondary

Assumptions

Transparency in the setting and evaluation of assumptions that may impact model application is of great importance in the planning and documentation of any model-informed drug discovery and development (MID3) activity.[1]

[1] EFPIA MID3 Workgroup, Marshall SF, Burghaus R, Cosson V, Cheung S, Chenel M, DellaPasqua O, Frey N, Hamrén B, Harnisch L, Ivanow F, Kerbusch T, Lippert J, Milligan PA, Rohou S, Staab A, Steimer JL, Tornøe C, Visser S. Good Practices in Model-Informed Drug Discovery and Development: Practice, Application, and Documentation. *CPT Pharmacometrics Syst Pharmacol.* 2016; **5**(3):93-122. doi: 10.1002/psp4.12049. Epub 2016 Mar 14

Assumptions are documented using a structured ASCII text file, using fields as defined below:

Field name	Meaning	Possible values
Type	The classification of the assumption	pharmacological / physiological / disease / data / mathematical/statistical
AssumptionBody	The assumption itself	Free text (typically 1-2 sentences)
Justification	The justification for making the assumption	Free text (typically 1-2 sentences)
Established	Is the assumption new, or has it been previously established?	new / established
Testable	Is the assumption testable?	Free text (typically 1-2 sentences)
TestApproach	How to test the impact of the assumption	Free text (typically 1-2 sentences)
TestOutcome	How to evaluate the outcome of the testing of the assumption	Free text (typically 1-2 sentences)

An example in XML:

```
<?xml version="1.0"?>
<Assumption>
      <Type>Pharmacological</Type>
      <AssumptionBody>Emax model fixed to 100% is a more
physiological description of the data compared to a linear
model.</AssumptionBody>
      <Justification>Emax model is not better than linear
model; however, for this drug class, Emax of 100% is more
realistic.</Justification>
      <Established>New</Established>
      <Testable>Testable with a wider range of concentrations
(external/future study).</Testable>
      <TestApproach>
Comparison of simulated metrics
of interest between the
two competing models.
      </TestApproach>
      <TestOutcome>To achieve a 90% response
(assumed to be clinically
meaningful) requires a twofold
higher dose using the
Emax model compared to
the linear model.</TestOutcome>
</Assumption>
```

In order to support querying and reasoning to “assumption” entities, the following fields will be used. Note, these will be namespaced to “mid3”:

Metadata field name	Possible values
mid3:assumptionType	pharmacological physiological disease data mathematical
mid3:established	new established
mid3:testable	true false

Decisions

Decisions (model selection, based on outputs from assumption testing, and similar) are crucial to document. Decisions are defined as entities which physically take the form of simple ASCII text files containing 1-2 sentences describing the decison made.

```
<?xml version="1.0"?>
<Decision>
   This is the base model to be carried forward into the
stepwise covariate analysis.
</Decision>
```

Entity Relationships

Models, scripts, and data are created and change over time. We wish to capture the relationships between these types of entities so that we understand where “they came from”.

We have 4 types of “derivation” available to us. Note however that we can attach metadata to a relationship in order to impart extra meaning.

The following relationships are available:

- “derived from” - is a transformation of an entity into another, an update of an entity resulting in a new one, or the construction of a new entity based on a pre-existing entity
- “revision of” - a derivation for which the resulting entity is a revised version of some original
- “quotation” - a quotation is the repeat of (some or all of) an entity, by someone who may or may not be its original author.
- “primary source” - A primary source for a topic refers to something produced by some agent with direct experience and knowledge about the topic, at the time of the topic's study, without benefit from hindsight.

An extra relationship is also available - “wasInfluencedBy”. This is a general relationship that can be used to capture a link between **any object in the system** (i.e. between agents, entities and activities). All other relationships (e.g. used, generated, derived from, etc.) are specialisations of this relationship.

We will use these relationships as follows:

Relationship	**Usage**

derived from	Capturing parent/child relationships between models
revision of	Capturing updates/new versions of existing files. This replaces the previous version of the entity (i.e. the previous version is no longer “in” the project)
quotation	Not planned to be used, although it may be useful for QC purposes as there is not change to the original However, semantically, “quotation” sounds like something else.
primary source	Would be used when a model is imported into a project from a model library
was influenced by	A weak relationship exists between two entities; a change in the “influencer” will not necessarily affect the “influencee”

Relationship Metadata

The following types of metadata will be attached to relationships to provide extra information about the meaning of the relationship.

Metadata field name	**Values**	**Meaning**
prov:type	ddmore:Specialisation	Used to denote that a model is a child of another model.

Activities

An “activity” is something that occurs over a period of time and acts upon or with entities; it may include consuming, processing, transforming, modifying, relocating, using, or generating entities.

In a typical pharmacometrics modelling project, the following are examples of activities undertaken by the modeller that we would wish to capture

- cloning a model
- performing a parameter estimation
- performing a simulation
- performing an SCM
- QCing a model
- making an assumption
- taking a decision
- updating an entity’s description (i.e. the metadata that describes an entity within the workflow system)

In order to allow the client to request the most appropriate information, depending on requirements, the following metadata will be used to describe each activity:

Metadata field name	**Value**	**Meaning**
prov:type	ddmore:clone	

prov:type	ddmore:estimate	
prov:type	ddmore:simulate	
prov:type	ddmore:QC	
prov:type	ddmore:decision	
prov:type	ddmore:assumption	

5 User actions to capture

Principles

The mechanism by which we capture the actions taken by the user is based on the following principles:

1. An entity referenced within the workflow database must be persistent. This means that is must be stored within a reliable, secure, referenceable and perpetual system - for instance a version control system.
2. All records stored within the database are **immutable** - i.e. they cannot be changed once they are stored. If a change it necessary, then it should be captured as a new record, which replaces the previous one.

Scenarios

The following scenarios summarize the actions that a user may perform on entities, and indicate how they will be captured in PROV-N.

Template models

Model A exists within a central repository.

Model B is a new file that is added into the project, by directly copying model A.

```
document
  default <http://www.ddmore.eu#>
  central
<http://wwwdev.ebi.ac.uk/biomodels/model-repository#>
```

```
  repo <https://github.com/johndoe/examplerepo.git#>

  entity(central:modelA)
  entity(repo:modelB, [ prov:type="ddmore:model",
prov:location="/models/modelB.ctl”] )

  wasDerivedFrom(central:modelA, repo:modelB,
[prov:type='prov:PrimarySource' ])

end document
```

Create child model

Model P exists within a project.

Model Q is a child model of Model P.

```
  entity(repo:modelP, [ prov:type="ddmore:model",
prov:location="/models/modelP.ctl”] ))
  entity(repo:modelQ, [ prov:type="ddmore:model",
prov:location="/models/modelQ.ctl”] )
  wasDerivedFrom(repo:modelQ, repo:modelP,
[prov:type=”ddmore:Specialization”] )
```

Weak links between models

Model P exists within a project.

Model Q is has influenced the development of Model P, but is not a child.

```
  entity(repo:modelP, [ prov:type="ddmore:model",
prov:location="/models/modelP.ctl”] ))
  entity(repo:modelQ, [ prov:type="ddmore:model",
prov:location="/models/modelQ.ctl”] )
  wasInfluencedBy(repo:modelQ, repo:modelP )
```

Updating a file

A user alters the content of a file..

That version of the file is no longer available in the latest version of the “project”.

As the previous file is not there any more, it should be captured as a revision.

```
  entity(repo:modelA, [ prov:type="ddmore:model",
prov:location="/folder/modelA.ctl”] ))
  entity(repo:modelA1, [ prov:type="ddmore:model",
prov:location="/folder/modelA.ctl”] )
  wasDerivedFrom(repo:imodelA, repo:modelA1,
[prov:type='prov:Revision'])
```

See the solution design for more details on entity ids.

Moving a file

A user moves a file within a project into a different location.

The file is otherwise unchanged. The file is no longer available at the previous location.

As the previous file is not there any more, it should be captured as a revision.

```
  entity(repo:modelA, [ prov:type="ddmore:model",
prov:location="/old/folder/modelA.ctl”] ))
  entity(repo:modelA1, [ prov:type="ddmore:model",
prov:location="/new/folder/modelA.ctl”] )
  wasDerivedFrom(repo:modelA, repo:modelA1,
[prov:type='prov:Revision'])
```

Adds metadata to an Entity

In this case, the user wishes to add some description to the Entity, for instance:

- update the label (“final”, “base”, <none>)
- update the QC status (“true”, “false”)
- update its pivotal status (“true”, “false”)

According to principle 2, an Entity record should be immutable in the workflow database. Therefore this information should be captured as a new version (aka “a revision of”) the previous entity, replacing it in the “live” version of the workflow.

QC Status

Consider an entity previously stored within the worfkflow database:

```
 entity(repo:modelA, [ prov:type="ddmore:model",
prov:location="/folder/modelA.ctl”] ))
```

This model has been reviewed and will be updated with a QC status.

While this is similar to the scenario above it may be useful to capture the QC activtiy. There is a transformation that is taking place, to convert the entity from a non-qc'ed entity into a qc'ed one.

It is possible to reference an activity in a "wasDerivedFrom" relationship. Therefore the "QC" itself can be captured as an Activity, as shown:

```
entity(Phylinda, [prov:type='prov:Person', name="Phylinda
Chan", userid="pchan"])
entity(run1.ctl, [prov:location="/folder/run1.ctl"])
activity(qcprocess1, [prov:type="QC"])
wasAssociatedWith(qcprocess1, Phylinda)
entity(run1qc.ctl, [prov:location="/folder/run1.ctl",
ddmore:qcStatus="true"])
wasDerivedFrom(run1.ctl, run1qc.ctl, qcprocess1, outputId1,
inputId1)
wasGeneratedBy(outputId1; run1qc.ctl, qcprocess1, -)
used(inputId1; qcprocess1, run1.ctl, -)
```

Which says that:
- Phylinda did a QC. An input was the run, and it generated a qc'ed run

An Activity can generate other outputs - so there could also be a QC Report generated from this, or a series of Decision entities.

There could also be multiple models QCed at the same time (many models as input) and many outputs (many qc'ed models - with a true/pass or false/fail flag). This is ok, as we can map the non qc'ed model to the qc'ed model with many wasDerivedFrom statements, that reference the same activity.

Labelling

The same pattern could be applied to capture labelling and other demarcations.

Consider an entity previously stored within the worfkflow database:

```
 entity(repo:abc-123, [ prov:type="ddmore:model",
prov:location="/folder/modelA.ctl”] ))
```

This will be updated to contain the new label “final”.

```
 activity(labellingActivity1234, [prov:type="label"])
 entity(repo:abc-654, [ prov:type="ddmore:model",
prov:location="/folder/modelA.ctl”, prov:label=”Final”] ))
 wasDerivedFrom(repo:abc-654, repo:abc-123,
labellingActivity1234, outputId1, inputId1)
 wasGeneratedBy(repo:abc-654, labellingActivity1234, -)
 used(labellingActivity1234, repo:abc-123, -)
```

Note that the entity ID is different, even though it is identical to the previous revision in all respects other than it has some extra description. This is required so that we can differenticate it from the previous revision.

This would also support activities as marking something as pivotal.

- Tests various covariates (covariate model(s) "wasCovariateModelOf" base model, under "wasRevisionOf"). Here, it is also possible define the relationships of covariate models relative to each other, for example covariate model 1 "wasVariantOf" (also under "wasRevisionOf") covariate model 2, but both "wasCovariateModelOf" base model.
- Develops final model from covariate models ("wasFinalModelOf" under "wasRevisionOf")

Querying the Workflow Store

Solution Design

Version Control System

Bundles

A Document contains a number of Provenance descriptions that are used to describe the provenance information.

A Document can contain zero or many Bundles that contain provenance descriptions.

A Bundle can also have provenance information attached to it.

We use a Bundle to

Document Revision History

Version	Author	Date	Description