Tools and techniques for creation, use and management of information models as metadata for system-of-systems interoperability

Edition 2.0

Rob Atkinson, William Francis, Nick Murray

June 2012

Revised Stephen Richard (remove template text from CSIRO)

April 2018

A water information R & D alliance between the Bureau of Meteorology and CSIRO's Water for a Healthy Country Flagship

Water for a Healthy Country Flagship Report series ISSN: 1835-095X

Citation

Atkinson R., Francis, W., Murray,N , 2012. *Tools and techniques for creation, use and management of information models as metadata for system-of-systems interoperability. Edition 2.0 [Draft].*
CSIRO: Water for a Healthy Country National Research Flagship

Copyright and disclaimer

Important disclaimer

Important disclaimer

NOTE—this report was apparently never formally completed or released, but remains the only known documentation for the SolidGround Enterprise Architect extension, which includes useful functionality for generating XML schema from UML models in the ISO TC211 workflow. It is thus included with the installer for that extension as is

Foreword

This document provides an updated overview of tools and techniques emerging from an ongoing research agenda into description of information system interoperability aspects. These concerns have many applications, and the tools generated cover many different aspects of these problems. This report is intended as a companion to the Solid Ground set of tools, to assist users to orient themselves according to the underlying problems being addressed and the behaviour of the tools.

Contents

Foreword i

Acknowledgments iv

Model Registry functions 6

1.2 Registering model components with dependencies 6

1.3 Browse model registry 9

2 Standards conformance 13

2.1 Assign sequence number 13

2.2 Generate package dependencies diagrams 15

2.3 Run Conformance tests 15

2.4 Generate class diagrams 17

2.5 Generate class context diagram 18

2.6 Clean duplicate tagged values 19

2.7 Validate mapping model 19

3 Other Solid Ground functions 20

3.1 Reverse engineering a model from an ArcGIS geodatabase and creating a conceptual model 20

3.2 Reverse engineering a model from a database and creating a conceptual model 21

3.3 Refactoring existing models using the *Remove System Metadata* tool 26

3.4 Generate Platform-Specific Model (UML to database) 28

3.5 Generate Data Migration Script 31

Figures

Figure 1: Solid Ground Registry Browser 10
Figure 2: Displaying the attributes within the elements of a particular model 10
Figure 3: Imported classes prior to refactoring 23
Figure 4: Attributes dialogue 24

Tables

Table 1: Solid Ground Registry Browser components 10
Table 2: ISO equivalents of imported (ie database-dependent) types 24

Acknowledgments

The Authors wish to thank the Bureau of Meteorology’s Water Division under the Water Information Research and Development Alliance (WIRADA) and all the reviewers for their effort and valuable input.

Model Registry functions

1.2 Registering model components with dependencies

1.2.1 RATIONALE

You wish to register a conceptual model with the Solid Ground Model Registry so that:

- components of the model can be reused by other models;
- model component metadata is preserved; and
- there is an explicit contract between the i) the conceptual model and models depending on it, and ii) the conceptual model and models it depends upon.

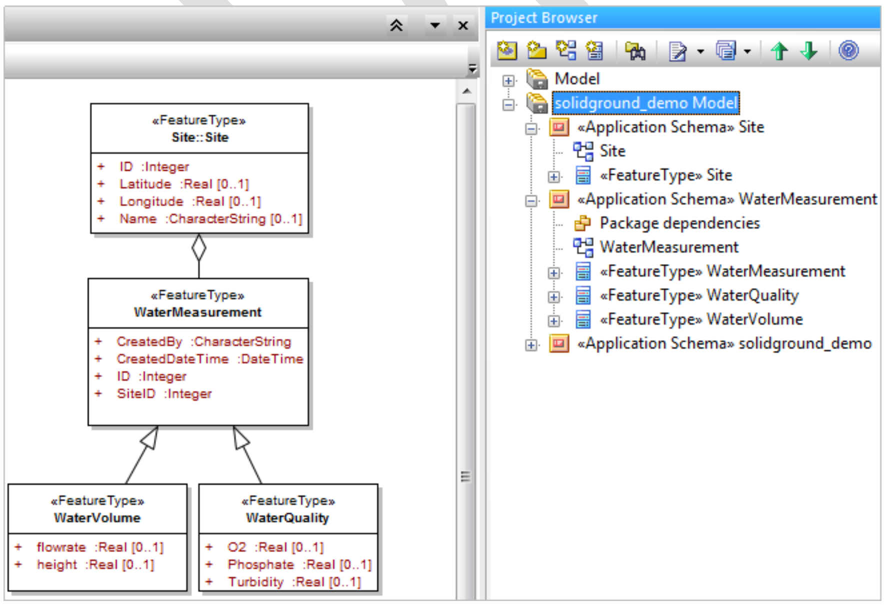
The conceptual model you wish to register optionally (but usually) has dependencies on other models.

Once registered, you can use Solid Ground to visualise dependencies to and from the model and import dependencies as required (this is covered in Scenario 3).

1.2.2 PRECONDITIONS

- The conceptual model exists in a stand-alone EA project file or shared modelling repository (ie not the Solid Ground registry, but a server-based, multi-user EA modelling repository).
- The conceptual model conforms to the ISO 19103 Conceptual Schema Language (uses standard definitions for modelling elements).
- The model may already have been designed from the "top-down" or refactored from one or more imported, pre-existing implementation models, as in the previous section.
- The conceptual model's package must have a valid version number, comprising three parts x.y.z, where x, y and z are integers.
- Each model that the current conceptual model depends on must have a package dependencies diagram. See page 15.

For the example, the following model is assumed:



1.2.3 INSTRUCTIONS

1. Take all classes and modelling elements that form a distinct and granular conceptual model and move them into their own package.

   In this example we will use the *WaterMeasurement* model from the previous scenario. ##Figure ref.

2. Add the <<ApplicationSchema>> stereotype to this package. If you have imported a model from ArcGIS or from a database, the package may already have the <<Application Schema>> stereotype.

3. Delete any existing Package Dependencies diagrams from the model. (It may already have one if you are refactoring an existing model, or have imported the model from a database).

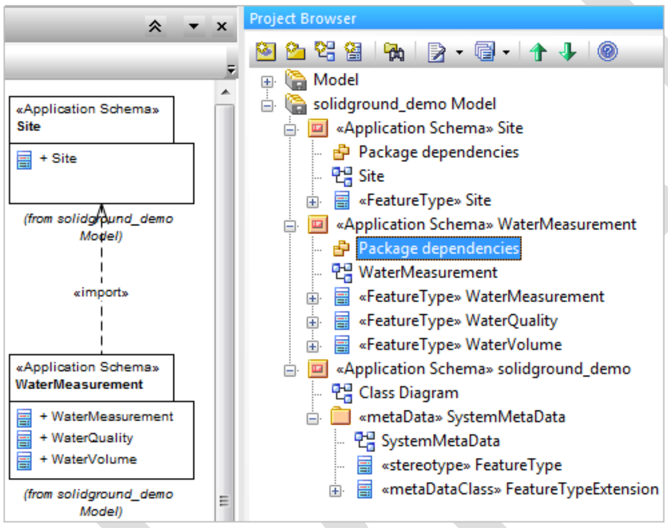
   Will SG do this? Does it matter if you have > 1 package dependency diagram?

4. Ensure **all** packages in the model you wish to register have Package Dependencies diagrams.

   Right-click the package(s) and select *Extensions > Solid Ground > Standards Conformance > Generate Package Dependencies Diagram*.

   Prior to the June 2012 release of Solid Ground, you had to use one of the Hollow World tools to generate the Package Dependency Diagram. This particular tool is now part of Solid Ground.

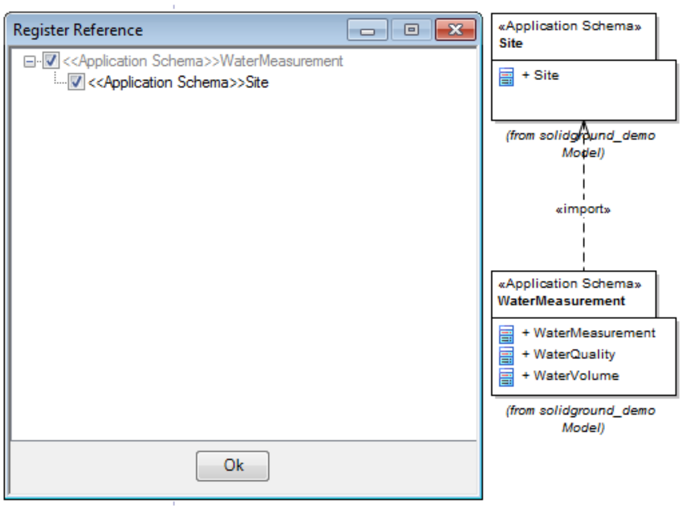
   For the *WaterMeasurement* package, the package dependency diagram should appear similar to this:



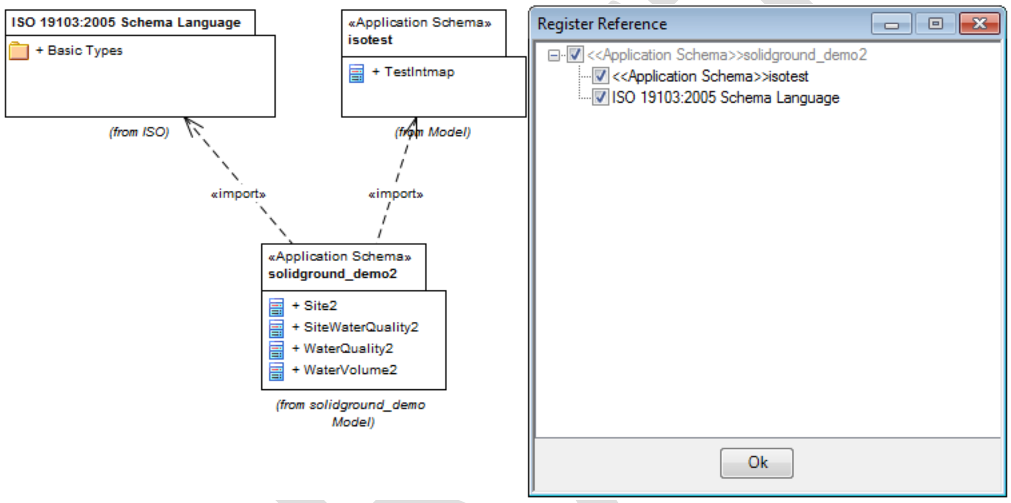
«Application Schema»
Site
+ Site
(from solidground_demo Model)
«import»
«Application Schema»
WaterMeasurement
+ WaterMeasurement
+ WaterQuality
+ WaterVolume
(from solidground_demo Model)
Project Browser
Model
solidground_demo Model
«Application Schema» Site
Package dependencies
Site
«FeatureType» Site
«Application Schema» WaterMeasurement
Package dependencies
WaterMeasurement
«FeatureType» WaterMeasurement
«FeatureType» WaterQuality
«FeatureType» WaterVolume
«Application Schema» solidground_demo
Class Diagram
«metaData» SystemMetaData
SystemMetaData
«stereotype» FeatureType
«metaDataClass» FeatureTypeExtension

5. To register the model with the Solid Ground registry, in the EA Project Browser, right-click the package (here, *WaterMeasurement* ##figref) and select *Extensions > Solid Ground > Model Registry > Register Model*.

6. At the Register Reference dialogue, tick the box next to all models you wish to register. For the *WaterMeasurement* example, the Register Reference dialogue should show the dependencies, as follows:



If the model contains dependencies that have *not* been registered previously, Solid Ground will display them in this dialogue and select them, so they will registered automatically. For example:



Here, the *solidground_demo2* model depends on ISO19103, but also another package, *isotest*. As *isotest* has not been registered, Solid Ground offers to register it for you.

7. Solid Ground will tell you if any of the dependent packages have been registered previously. you may optionally replace them.

1.2.4 POSTCONDITIONS

The model and any model it depends upon will be registered in the Solid Ground registry, with metadata containing the version, creator (author), creation date and governance. The model is now available for other modellers to "import" and reference in their models.

1.2.5 POINTS TO REMEMBER

Solid Ground ensures that dependency relationships are maintained. However, Solid Ground does not support circular dependencies, as they are not defined in ISO 19100

There may be erroneous dependencies in your models (particularly if they are haven't been reviewed in detail) and the registration process will help to identify any incorrect dependencies.

1.3 Browse model registry

1.3.1 RATIONALE

You have a modelling task, and wish to re-use existing models in order to:

- speed up the modelling development, testing and deployment process
- conform to ISO, OGC and (perhaps) organisational standards

1.3.2 PRECONDITIONS

Your organisation has a pre-existing deployment of the Solid Ground registry. This registry has been pre-populated with reference models (ISO, OGC etc) and perhaps "template" models developed by others in your organisation.

Installing, populating and deploying a Solid Ground registry is beyond the scope of this document.

1.3.3 INSTRUCTIONS

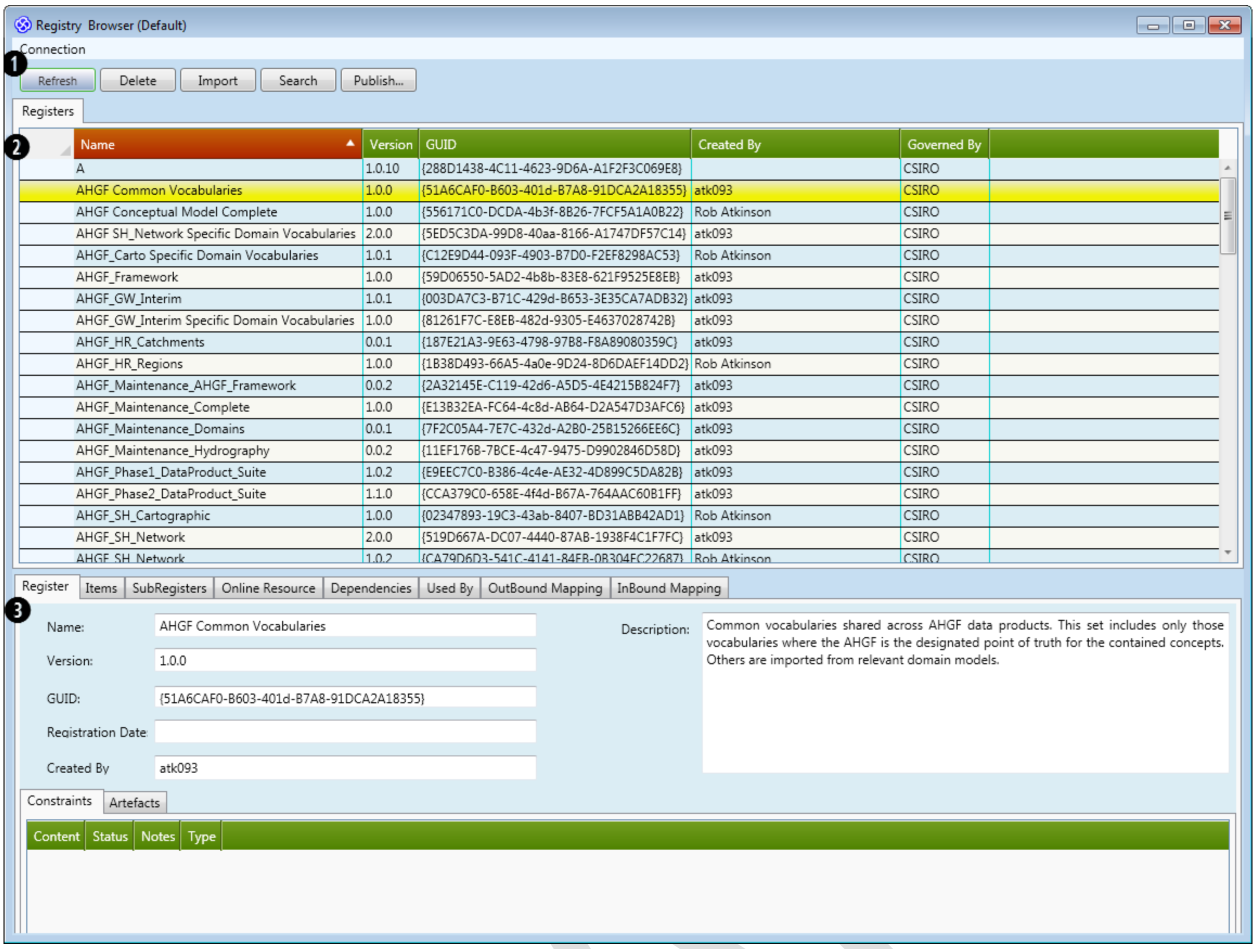
1. Create a new EA project.
2. In the EA Project Browser, right-click any package.
3. From the pop-up menu, select *Extensions > Solid Ground > Model Registry > Browse Model Registry*. The registry browser window appears.

   The top part of the screen, containing the Connection, Refresh, Delete, Import, Search and Publish buttons, lists actions you perform on the selected model. See ❶ in Figure 1.

   The *Registers* tab lists the models present in the current registry. Click any of the column headings to sort in ascending or descending order. See ❷ in Figure 1.

   The lower part of the screen (containing the Register, Items, Subregisters, etc tabs) lists various attributes of the currently-selected model. See ❸ in Figure 1.

   Table 1 describes each part of the Registry Browser in more detail.
4. To see the attributes of elements within a model, select the model, then the *Items* tab, then the *Attributes* tab. See Figure 2.

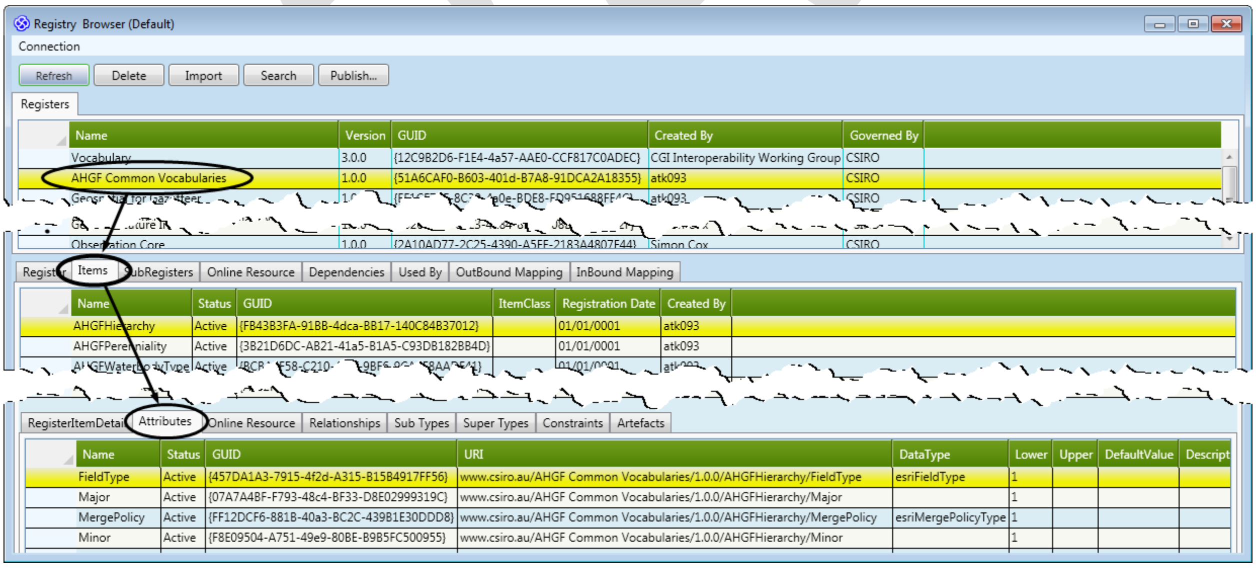


Name	Version	GUID	Created By	Governed By
A	1.0.10	{288D1438-4C11-4623-9D6A-A1F2F3C069E8}		CSIRO
AHGF Common Vocabularies	1.0.0	{51A6CAF0-B603-401d-B7A8-91DCA2A18355}	atk093	CSIRO
AHGF Conceptual Model Complete	1.0.0	{556171C0-DCDA-4b3f-8B26-7FCF5A1A0B22}	Rob Atkinson	CSIRO
AHGF SH_Network Specific Domain Vocabularies	2.0.0	{5ED5C3DA-99D8-40aa-8166-A1747DF57C14}	atk093	CSIRO
AHGF_Carto Specific Domain Vocabularies	1.0.1	{C12E9D44-093F-4903-B7D0-F2EF8298AC53}	Rob Atkinson	CSIRO
AHGF_Framework	1.0.0	{59D06550-5AD2-4b8b-83E8-621F9525E8EB}	atk093	CSIRO
AHGF_GW_Interim	1.0.1	{003DA7C3-B71C-429d-B653-3E35CA7ADB32}	atk093	CSIRO
AHGF_GW_Interim Specific Domain Vocabularies	1.0.0	{81261F7C-E8EB-482d-9305-E4637028742B}	atk093	CSIRO
AHGF_HR_Catchments	0.0.1	{187E21A3-9E63-4798-97B8-F8A89080359C}	atk093	CSIRO
AHGF_HR_Regions	1.0.0	{1B38D493-66A5-4a0e-9D24-8D6DAEF14DD2}	Rob Atkinson	CSIRO
AHGF_Maintenance_AHGF_Framework	0.0.2	{2A32145E-C119-42d6-A5D5-4E4215B824F7}	atk093	CSIRO
AHGF_Maintenance_Complete	1.0.0	{E13B32EA-FC64-4c8d-AB64-D2A547D3AFC6}	atk093	CSIRO
AHGF_Maintenance_Domains	0.0.1	{7F2C05A4-7E7C-432d-A2B0-25B15266EE6C}	atk093	CSIRO
AHGF_Maintenance_Hydrography	0.0.2	{11EF176B-7BCE-4c47-9475-D9902846D58D}	atk093	CSIRO
AHGF_Phase1_DataProduct_Suite	1.0.2	{E9EEC7C0-B386-4c4e-AE32-4D899C5DA82B}	atk093	CSIRO
AHGF_Phase2_DataProduct_Suite	1.1.0	{CCA379C0-658E-4f4d-B67A-764AAC60B1FF}	atk093	CSIRO
AHGF_SH_Cartographic	1.0.0	{02347893-19C3-43ab-8407-BD31ABB42AD1}	Rob Atkinson	CSIRO
AHGF_SH_Network	2.0.0	{519D667A-DC07-4440-87AB-1938F4C1F7FC}	atk093	CSIRO
AHGF_SH_Network	1.0.2	{CA79D6D3-541C-4141-84EB-0B304FC22687}	Rob Atkinson	CSIRO

Name:	AHGF Common Vocabularies
Version:	1.0.0
GUID:	{51A6CAF0-B603-401d-B7A8-91DCA2A18355}
Registration Date	
Created By	atk093
Description:	Common vocabularies shared across AHGF data products. This set includes only those vocabularies where the AHGF is the designated point of truth for the contained concepts. Others are imported from relevant domain models.

Content	Status	Notes	Type

Figure 1: Solid Ground Registry Browser



Name	Version	GUID	Created By	Governed By
Vocabulary	3.0.0	{12C9B2D6-F1E4-4a57-AAE0-CCF817C0ADEC}	CGI Interoperability Working Group	CSIRO
AHGF Common Vocabularies	1.0.0	{51A6CAF0-B603-401d-B7A8-91DCA2A18355}	atk093	CSIRO
Geospatial for Gazetteer	1.0	[illegible]	atk093	CSIRO
Observation Core	1.0.0	{2A10AD77-2C25-4390-A5FF-2183A4807F44}	Simon Cox	CSIRO

Name	Status	GUID	ItemClass	Registration Date	Created By
AHGFHierarchy	Active	{FB43B3FA-91BB-4dca-BB17-140C84B37012}		01/01/0001	atk093
AHGFPerenniality	Active	{3B21D6DC-AB21-41a5-B1A5-C93DB182BB4D}		01/01/0001	atk093
AHGFWaterbodyType	Active	[illegible]		01/01/0001	atk093

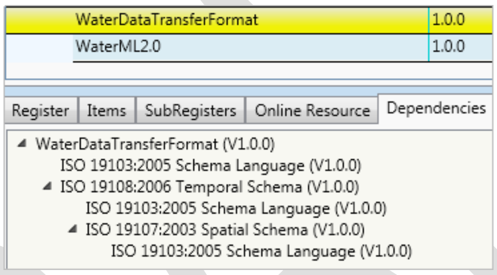
Name	Status	GUID	URI	DataType	Lower	Upper	DefaultValue	Descript
FieldType	Active	{457DA1A3-7915-4f2d-A315-B15B4917FF56}	www.csiro.au/AHGF Common Vocabularies/1.0.0/AHGFHierarchy/FieldType	esriFieldType	1			
Major	Active	{07A7A4BF-F793-48c4-BF33-D8E02999319C}	www.csiro.au/AHGF Common Vocabularies/1.0.0/AHGFHierarchy/Major		1			
MergePolicy	Active	{FF12DCF6-881B-40a3-BC2C-439B1E30DDD8}	www.csiro.au/AHGF Common Vocabularies/1.0.0/AHGFHierarchy/MergePolicy	esriMergePolicyType	1			
Minor	Active	{F8E09504-A751-49e9-80BE-B9B5FC500955}	www.csiro.au/AHGF Common Vocabularies/1.0.0/AHGFHierarchy/Minor		1			

Figure 2: Displaying the attributes within the elements of a particular model

Table 1: Solid Ground Registry Browser components

USER INTERFACE ELEMENT	PURPOSE
Upper panel of Registry Browser	
Connection	This is a drop-down menu allowing you to switch between registries. For example, you might have a "reference" registry that contains ISO, OGC, etc

USER INTERFACE ELEMENT	PURPOSE
	models and is read-only, and you may have your own "development" registry for developing and testing models.
Refresh	Refreshes and reloads the contents of the registry browser.
Delete	Deletes a model from the registry. **WARNING: This function deletes the selected model without confirmation**.
Import	Imports the selected model (and optionally all or some of the model's dependencies) into the currently-selected package within EA.
Search	Not yet functional in this release of Solid Ground.
Publish…	Publishes the selected model to a SKOS file.
Lower panel of Registry Browser	
Register	This tab displays basic metadata about the model selected in the Registers grid. Constraints & artefacts tabs What are these fields intended to show? Where does the content for these fields come from? Content Status Notes Type
Items	This tab shows each item (ie class, or element) within the model. This largely applies to controlled vocabularies, and is an experimental proof of concept that may be removed in future with the merging emphasis on publishing such data to a Linked data environment. RegisterItemDetail tab: shows metadata about the item (class or element) Attributes tab: shows metadata about the name and type of each attribute (property/field) within a class (or modelled element). Note the URI field, which points to the type definition of the item. Online Resource tab Relationships tab Sub Types tab Super Types tab Constraints tab Artefacts tab
Subregisters	Top panel: Name, URI, HarvestServer Middle panel: Intances: Name, Status, URI, Description Bottom panel: Properties: Name, Value

USER INTERFACE ELEMENT	PURPOSE
Online Resource	This tab displays a list of downloadable resources associated with the model. Solid Ground automatically creates an XMI and Solid Ground XML format of the model that you can download. You can also upload your own resources to share with others. Upload, Delete, Download What is an Application Profile, in the Upload Artifact dialogue? Name Application Profile Link Description Creation Date Created By
Dependencies	This tab shows a tree list of other models the selected model depends on, and the models those models depend on. This example shows the dependency tree for WDTF:  WaterDataTransferFormat 1.0.0 WaterML2.0 1.0.0 Register | Items | SubRegisters | Online Resource | Dependencies ◢ WaterDataTransferFormat (V1.0.0) ISO 19103:2005 Schema Language (V1.0.0) ◢ ISO 19108:2006 Temporal Schema (V1.0.0) ISO 19103:2005 Schema Language (V1.0.0) ◢ ISO 19107:2003 Spatial Schema (V1.0.0) ISO 19103:2005 Schema Language (V1.0.0)
Used By	Name, Version, GUID
OutBound Mapping	Import, Delete Name, Target Register, Version, GUID,Created By, Governed By
InBound Mapping	Import, Delete Name, Source Register, Version, GUID,Created By, Governed By

5. DeIn the EA Project Browser, right-click any package.

1.3.4 FROM THE POP-UP MENU, SELECT *EXTENSIONS > SOLID GRO*EPOSTCONDITIONS

1.3.5 POINTS TO REMEMBER

2 Standards conformance

2.1 Assign sequence number

2.1.1 RATIONALE

Attribute order is not important to UML but is very important to GML. Solid Ground provides a way of controlling order in UML by applying a *sequenceNumber* tagged value to each attribute and association connection role in an Application Schema. This ensures generated GML encodings are consistent.

This tool was previously available in the Hollow World EA Add-in, and is now available in Solid Ground.

2.1.2 PRECONDITIONS

You can use this tool only after you have selected a package with the stereotype "Application Schema".

This tool will apply sequence numbers only to classes that have stereotypes that are known to Solid Ground. Stereotypes are added via UML profiles

The current list of stereotypes is ???

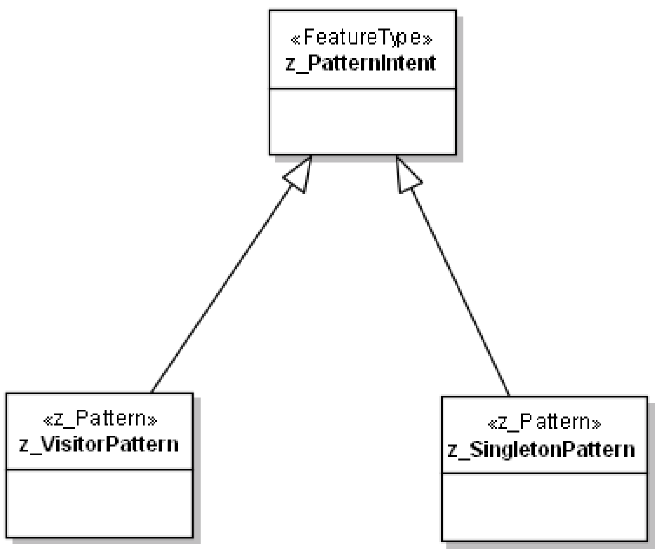
(RA: 11April12). Unsure how profiles are added.

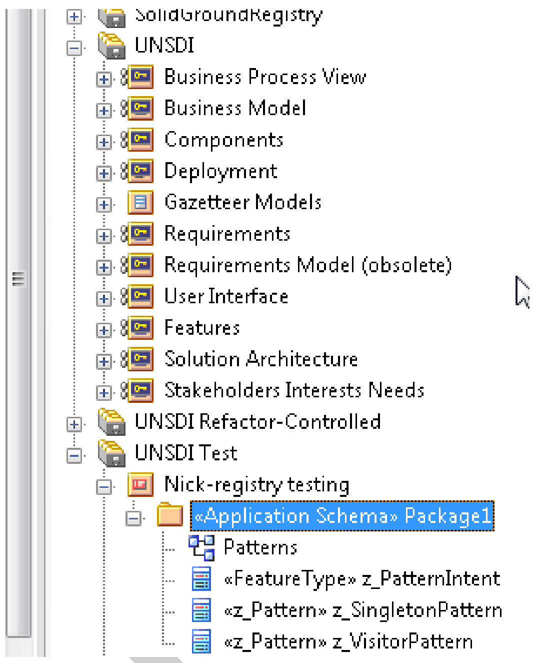
Once you have run this tool, you should not edit the tagged values manually unless you are sure the resulting order is what you want.

2.1.3 INSTRUCTIONS

6. Created a simple package stereotyped as << Application Schema>>.

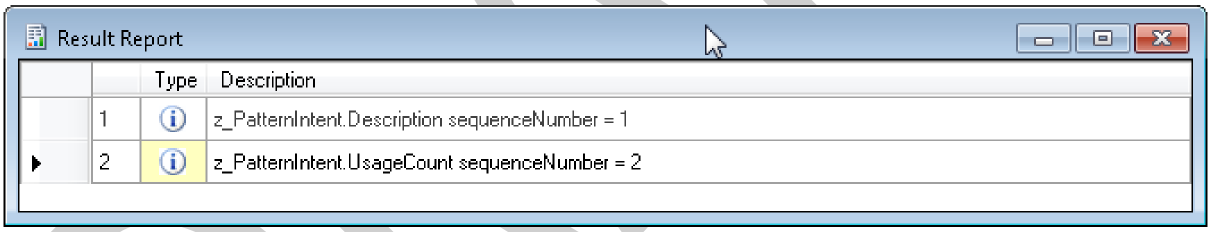
   ##redraw this diagram using Rob's models




SolidGroundRegistry
UNSDI
Business Process View
Business Model
Components
Deployment
Gazetteer Models
Requirements
Requirements Model (obsolete)
User Interface
Features
Solution Architecture
Stakeholders Interests Needs
UNSDI Refactor-Controlled
UNSDI Test
Nick-registry testing
«Application Schema» Package1
Patterns
«FeatureType» z_PatternIntent
«z_Pattern» z_SingletonPattern
«z_Pattern» z_VisitorPattern

7. Right-click the package.
8. From the pop-up menu, select *Extensions > Solid Ground > Standards Conformance > Assign Sequence Number.*
9. If successful, the message "The 'Assign Sequence Number' function has been executed successfully" appears. Solid Ground also displays a report of the results.

Re-capture this image



	Type	Description
1	ⓘ	z_PatternIntent.Description sequenceNumber = 1
2	ⓘ	z_PatternIntent.UsageCount sequenceNumber = 2

2.1.4 POSTCONDITIONS

Each attribute and association connector has a tagged value containing an

Q: Do the values monotonically increase for every attribute/connector in the selected package?

Q: Do the values assigned to the tagged values have any intrinsic meaning or are they present simply to provide an ordinal value?

2.1.5 POINTS TO NOTE

To edit a tagged value manually, right-click a class or connector, select *Attributes…* from the pop-up menu, then select Tagged Values on the left-hand side.

2.2 Generate package dependencies diagrams

2.2.1 RATIONALE

Creates a new diagram containing a list of packages the selected package depends on. Solid Ground uses the package dependency diagram to trace model dependencies.

This tool was previously available in the Hollow World EA Add-in, and is now available in Solid Ground.

If you try to register a model, and one or more of the dependent models does not have a Package Dependencies Diagram, run this tool to create the diagram.

2.2.2 REQUIREMENTS

You can use this tool only after you have selected a package with the stereotype <<Application Schema>>.

2.2.3 INSTRUCTIONS

1. In the EA Project Browser, right-click the package.
2. From the pop-up menu, select *Extensions > Solid Ground > Standards Conformance > Generate Package Dependencies Diagrams.*
3. If successful, the a new diagram appears in the editor and in the Project Browser (below the package). The diagram contains a package that in turn contains a list of dependencies.
4. For an example of a package dependency diagram, see section 1.2.3 on page 7.

2.3 Run Conformance tests

2.3.1 RATIONALE

Tests a package against a set of ISO-derived "best practice" modelling practices, and creates a report listing the elements (packages, associations, etc) that do not comply with these practices.

Some of the conformance checks are intended to check models intended for GML outputs; however you can use them to check for general model "well-formedness".

This tool was previously available in the Hollow World EA Add-in, and is now available in Solid Ground.

2.3.2 PRECONDITIONS

The conformance tests apply only to models using an ISO application schema; hence you can use this tool only after you have selected a package with the stereotype <<Application Schema>>.

2.3.3 INSTRUCTIONS

1. In the EA Project Browser, right-click the package.

2. From the pop-up menu, select *Extensions > Solid Ground > Standards Conformance > Run Conformance Tests.*

2.3.4 POSTCONDITIONS

Solid Ground displays a results report for each package, attribute and connector in the model. The report contains:

- Errors, where the model element does not pass one or more of the conformance checks; and
- Warnings, where the model element is missing some useful, but not essential, information. Examples are role names on associations, or Notes, in the model element's *Notes* field.

2.3.5 POINTS TO NOTE

The conformance check s are maintained as XQuery source files (.xq) in the SVN folder http://projects.arcs.org.au/svn/fullmoon/trunk/rules/conformance-test/ISO19136-V3.2-AnxE_XMI-V1.1 and are listed in the table below.

These checks don't include warnings eg associations having role-names, or elements having non-null values for the *Notes* field.

TEST NO	TEST
1	The model must have least one outermost UML package with the stereotype <<Application Schema>>.
2	The model must have only one UML package with the stereotype <<Application Schema>>. In other words, you must not have Application Schemas within Application Schemas. In the following tests, this package is denoted the "target Application Schema package".
3	The model must have at least one publicly visible UML class within the target Application Schema package.
4	The target Application Schema package must have non-null values for both the *targetNamespace* and *xmlns* tagged values.
5	All Application Schema and Leaf packages within the model must have an *xsdDocument* tagged value.
6	All *xsdDocument* tagged values in Application Schema and Leaf packages within the model must be non-null and unique.
7	All *@xmi.id* values in the XMI file must be unique. An apparent bug in EA export can duplicate tagged values for the root package. It may not be a 'bug' per se as it seems to have appeared in the GeoSciML model only after manual user errors with synchronisation of dependent packages. This test is included in order to eliminate duplicate values.
8	All *EAStub* names within the model must be unique.
9	Not defined.
10	Each name from *EAStub* elements must have corresponding records in registers or type-mapping tables depending on *EAStub/@UMLType*.
11	All attributes and association ends of classes must have a *sequenceNumber* tagged value.
12	In a model, all instances of the *sequenceNumber* tagged value must be unique across all attributes and association ends of classes.
13	All attribute and association target types are either internal or external classes, or EA XMI IDs that correspond to the NULL-type. Sorry I have NO idea what this one means!
14	If the model contains classes stereotyped with <<CodeList>> and these classes have the tagged value

TEST NO	TEST
	asDictionary="true", then the classes must also contain the *codeSpace*, *dictionaryIdentifier*, and *memberIdentifierStem* tagged values.
15	Checks for incorrect use of <<DataType>>, <<CodeList>> or <<Enumeration>> classes in the model. Classes of these stereotypes should only be used in the following contexts: 1. Attribute types; 2. DataType classes may be the source of any association; 3. Target of a composition association, navigable towards the <<DataType>>, <<Enumeration>> or <<CodeList>>. Use in any other association is an error.
16	Any package with the stereotype <<Leaf>> shall not contain another package.
17	All classes in the model shall have a suitable stereotype.
18	Every tagged value in the model is applied to the correct model element type and stereotype.
19	For any element in the model having tagged values, there shall be no more than one tagged value with the same tag (ie same name) on that element.
20	Asserts datatypes of tagged values.
21	Not defined.
22	Each navigable association-end must have a role-name (in order to be encoded).
23	Association model elements shall not have **both** ends set to "non-navigable; where the tagged value *isNavigable="false".*
24	Not defined.
25	The input model must use XMI version 1.1.
26	Composition associations shall not be tagged: *inlineOrByReference="byReference"* or *inlineOrByReference="inlineOrByReference".*
27	All targeted association-ends having the tagged value *byReference* must have a specified identity. ie the association end must be stereotyped as one of <<FeatureType>>, <<Type>>, <<Union>>, <<featureType>>, <<type>>, <<union>> or no-stereotype, where the tagged value is "" or ().

2.4 Generate class diagrams

2.4.1 RATIONALE

In EA, a model is defined using relationships that exist in its underlying database. Models do not "exist" on a class diagram: when creating a model, you can use class diagrams to capture the relationships between model elements, then you can delete the diagram: this keeps the model tidy and stops proliferation of half-completed diagrams that add no value to the model.

##Ref to Stephen's docs: models need to do work.

At a later time, if you need to see a class diagram you can re-create one using the Solid Ground Generate Class Diagrams tool.

This tool was previously available in the Hollow World EA add-in, and is now available in Solid Ground.

2.4.2 PRECONDITIONS

You must have an EA model containing classes and, optionally, relationships between those classes.

Neither the classes nor the package containing the classes needs to be stereotyped.

2.4.3 INSTRUCTIONS

1. In the EA Project Browser, right-click the package.
2. From the pop-up menu, select *Extensions > Solid Ground > Standards Conformance > Generate Class Diagram.*

2.4.4 POSTCONDITIONS

EA creates a new diagram containing the classes in the selected package, and also any relationships present.

2.5 Generate class context diagram

2.5.1 RATIONALE

This tool was previously available in the Hollow World EA add-in, and is now available in Solid Ground.

2.5.2 PRECONDITIONS

You must have an EA model containing classes and, optionally, relationships between those classes.

Neither the classes nor the package containing the classes needs to be stereotyped.

2.5.3 INSTRUCTIONS

1. In the EA Project Browser, right-click the package.
2. From the pop-up menu, select *Extensions > Solid Ground > Standards Conformance > Generate Class Context Diagram.*

2.5.4 POSTCONDITIONS

EA creates a new diagram containing the classes in the selected package, and also any relationships present.

2.6 Clean duplicate tagged values

2.6.1 RATIONALE

In earlier versions of EA and Solid Ground it was possible to create two tagged values with the same name on the one model element. This causes output problems, and so the Clean Duplicate Tagged Values tool was added to Solid Ground to remove the duplicates.

2.6.2 PRECONDITIONS

You must have an EA model containing classes and attributes and optionally, relationships.

2.6.3 INSTRUCTIONS

1. In the EA Project Browser, right-click the package.
2. From the pop-up menu, select *Extensions > Solid Ground > Standards Conformance > Clean Duplicate Tagged Values.*

2.6.4 POSTCONDITIONS

Any duplicate tagged values are removed.

2.7 Validate mapping model

2.7.1 RATIONALE

**prototype tool

2.7.2 PRECONDITIONS

2.7.3 INSTRUCTIONS

1. In the EA Project Browser, right-click the package.
2. From the pop-up menu, select *Extensions > Solid Ground > Standards Conformance > Validate Mapping Model.*

2.7.4 POSTCONDITIONS

3 Other Solid Ground functions

3.1 Reverse engineering a model from an ArcGIS geodatabase and creating a conceptual model

3.1.1 RATIONALE

This scenario uses the built-in EA "Import from ArcGIS Workspace" function, available in EA from versions 9.3.x and later.

See the EA help topic *Domain-based models > Geodatabase Design* for ArcGIS for more details.

3.1.2 PRECONDITIONS

In Arc Catalog, export the desired workspace as a schema only.

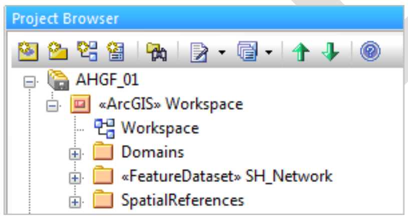
Example file: AHGF_01.xml

3.1.3 INSTRUCTIONS

1. In EA, select *Extensions > ArcGIS > Import ArcGIS Workspace XML*.
2. In the *Import Package from ArcGIS XML* dialog, select the XML file.
3. Click **Import**.
4. EA will ask if you want the imported model to be placed at the root level in the Project Browser, or under another package. Click Yes or No as required.

3.1.4 POSTCONDITIONS

EA populates the Project Browser.



See the EA help topic *Domain-based models > Geodatabase Design for ArcGIS > Import ArcGIS XML Workspace* for more details.

The Domains package contains

The *<<FeatureDataset>> SH_Network* package contains a UML model of features imported from the ArcGIS workspace. Each feature is modelled as a class stereotyped with its feature type.

The *SpatialReferences* package contains a class modelling the spatial referencing system used in the model. Each attribute is modelled as an EA tagged value.

3.1.5 POINTS TO NOTE

Extensions > *ArcGIS* > *Set Coordinate System* – is this necessary?

3.2 Reverse engineering a model from a database and creating a conceptual model

3.2.1 RATIONALE

An existing database represents an implicit data model, regardless of whether or not the database has – or had! – a formal schema. The database therefore represents a *potential* component in an interoperable system.

In the case of a domain without a formal model, a database contains the seeds of a domain model (limited to the operational scope of the database, but nevertheless valid). In this scenario, a modeller:

- reverse engineers one or more database sources to extract their schemas;
- refactors the reverse-engineered model to extract common concepts; and
- register the definitions of these concepts in a *conceptual model*, which should be free of redundancy and mutual dependencies between independently-governed components.

Semantics (concept definitions) in the form of "feature type models" (the *properties* or *attributes* of those concepts) are the key entities that need to be directly defined at design time.

Other examples in this document demonstrate how these can be later registered and published to support others subscribing to such reusable resources in a distributed system. This is seen as the long term future of an evolving system.

3.2.2 PRECONDITIONS

The model exists in a RDBMS. Solid Ground supports a range of database technologies, including MySQL, Postgres, Oracle and SQL Server.

3.2.3 INSTRUCTIONS

Overview

Specify the database location & type

Select the database dialect to UML mapping file

Import the model

Refactor/run system metadata tool etc

Invoke "update with dependencies" to create the links to the ISO types

Generate dependencies

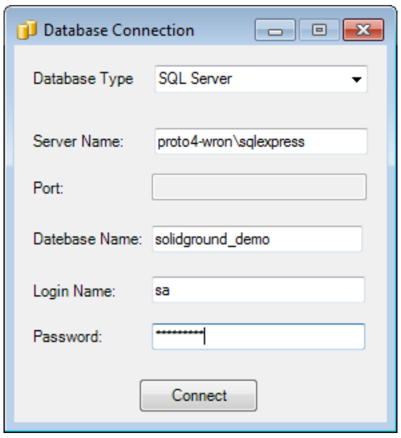
Add sequence numbers

Run conformance

Tweak model metadata

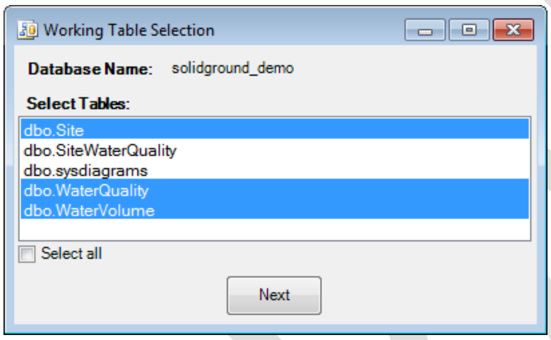
register as "model" .

1. In the EA Project Browser, select a model or package as the destination for the imported model: the imported model will appear underneath, or subordinate to, the selected model or package.
2. Right-click the model or package, then select *Extensions > Solid Ground > Import PSM tools > Database to UML.*
3. Select the database type and configuration.



Database Type	SQL Server
Server Name:	proto4-wron\sqlexpress
Port:	
Datebase Name:	solidground_demo
Login Name:	sa
Password:	**********

4. Select the relevant tables from the database, then click Next.



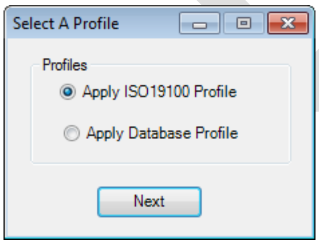
Database Name: solidground_demo

Select Tables:

dbo.Site
dbo.SiteWaterQuality
dbo.sysdiagrams
dbo.WaterQuality
dbo.WaterVolume

5. You now have two options:
   a Select *Apply ISO19100 Profile* to create a platform-independent model. Unlike other database reverse-engineering tools, Solid Ground is able to abstract a database schema using the ISO 19103 conceptual schema language, which saves a great deal of editing of the imported model.
   b Alternatively, select *Apply Database Profile* to create a platform-specific model.

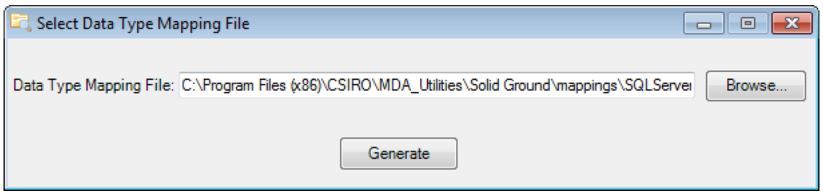
   This example uses a platform-independent model.



Apply ISO19100 Profile [✓]
Apply Database Profile

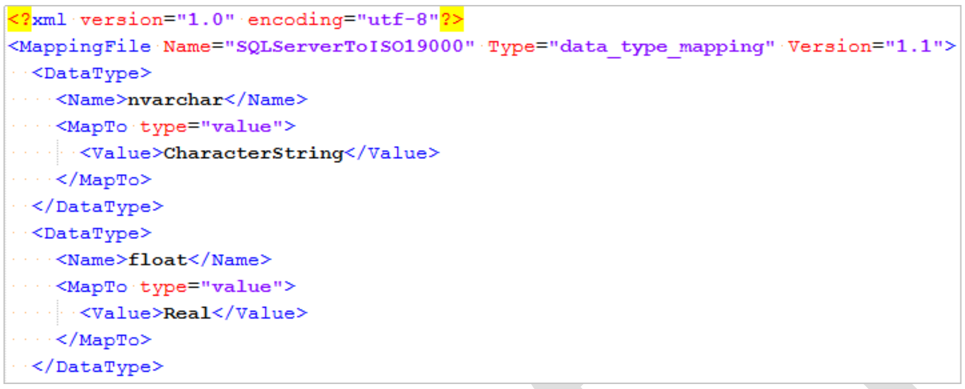
6. Select a "Data Type Mapping File".

   Solid Ground provides a suite of default mapping files under the installed directory, and will choose a sensible default based on the selected database type (ie Oracle, SQL Server, MySQL or Postgres).


Select Data Type Mapping File

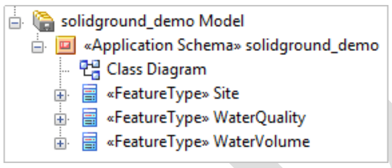
Data Type Mapping File: C:\Program Files (x86)\CSIRO\MDA_Utilities\Solid Ground\mappings\SQLServe

This mapping file defines how database elements are translated into UML elements. The figure below shows an extract of the SQL Server to ISO19100 mapping file, illustrating how the SQL types *nvarchar* and *float* are mapped to the ISO19100 vocabulary.


```xml
<?xml version="1.0" encoding="utf-8"?>
<MappingFile Name="SQLServerToISO19000" Type="data_type_mapping" Version="1.1">
  <DataType>
    <Name>nvarchar</Name>
    <MapTo type="value">
      <Value>CharacterString</Value>
    </MapTo>
  </DataType>
  <DataType>
    <Name>float</Name>
    <MapTo type="value">
      <Value>Real</Value>
    </MapTo>
  </DataType>
```

7. Click Generate.

EA imports the selected tables from the database and, using the ISO19100 profile, transforms the native database tables into classes stereotyped as <<FeatureType>> . The project browser will appear as follows:


```
solidground_demo Model
  «Application Schema» solidground_demo
    Class Diagram
    «FeatureType» Site
    «FeatureType» WaterQuality
    «FeatureType» WaterVolume
```

8. The class diagram appears as follows. This is a simplified example; real-life database-to-UML mappings will be more complex.

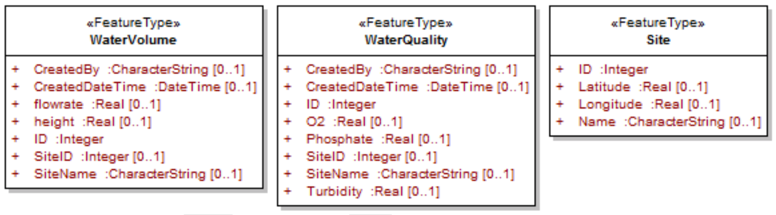


Figure 3: Imported classes prior to refactoring

You can now refactor the imported model elements into cleaner (and more re-usable) abstractions. This requires detailed knowledge of how the underlying domain-level concepts are applied in the database. There is no defined process for refactoring; it depends upon the nature of the underlying information model and how the database has been implemented. A simple refactoring example is provided in section 3.3 Refactoring existing models using the *Remove System Metadata*.

9. Note that the types in the classes in Figure 3 are primitives: Real, Integer and so on. This is acceptable in a database-dependent schema; however if you wish to conform to ISO standards, each primitive type must be mapped to its ISO equivalent.

The ISO types reside in the *ISO 19109:2005 Schema Language* model. You must map each of the types in the imported classes to its equivalent ISO type. Table 2 shows equivalents to some of the types imported in this example.

Note: Solid Ground is supposed to perform this mapping itself, matching names in the classes with names in the database-to-UML mapping file; however this is currently not working, and must be done manually.

Table 2: ISO equivalents of imported (ie database-dependent) types

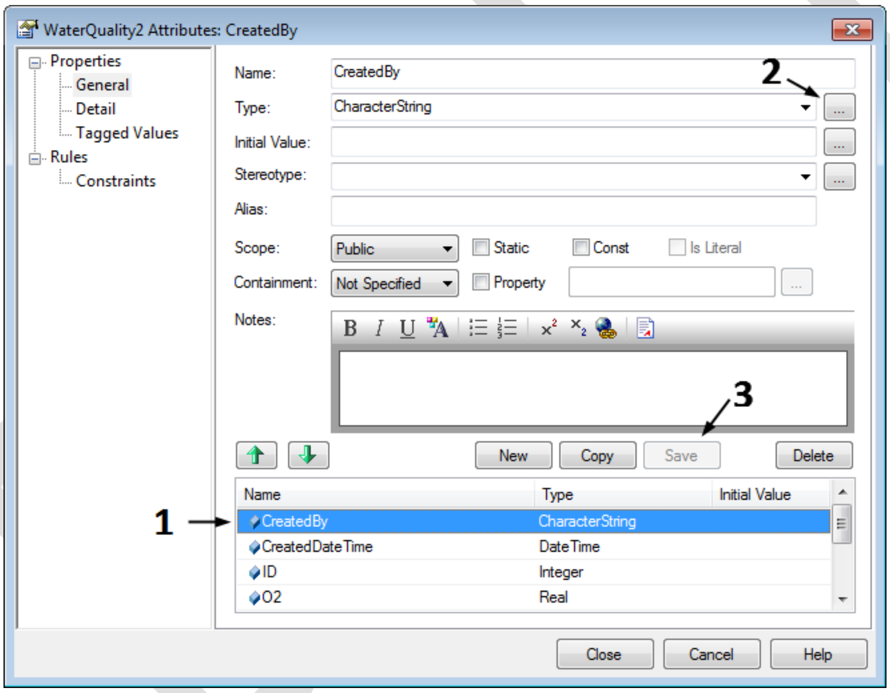
IMPORTED TYPE	ISO EQUIVALENT TYPE
CharacterString	ISO 19103:2005 - Basic Types - Text - <<Type>> CharacterString
DateTime	ISO 19103:2005 - Basic Types – Date and Time - <<Type>> DateTime
Real	ISO 19103:2005 - Basic Types - Numerics - <<Type>> Real
Integer	ISO 19103:2005 - Basic Types - Numerics - <<Type>> Integer

To perform the mapping (using the *WaterVolume* class as an example),

a In the EA project browser, or on the model class diagram, right-click *WaterVolume*, then select *Attributes*…

b Select an attribute. Example: *CreatedBy* : *Characterstring* (see (1) in Figure 4).

c Click the **Browse** button next to the Type field (see (2) in Figure 4).

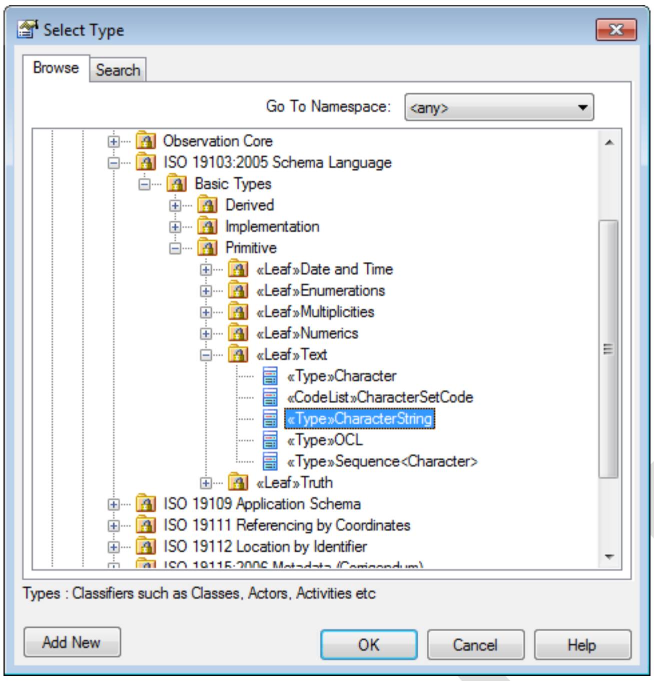

WaterQuality2 Attributes: CreatedBy

Name: CreatedBy
Type: CharacterString
Initial Value:
Stereotype:
Alias:
Scope: Public
Static
Const
Is Literal
Containment: Not Specified
Property
Notes:

Name	Type	Initial Value
CreatedBy	CharacterString	
CreatedDateTime	DateTime	
ID	Integer	
O2	Real	

Figure 4: Attributes dialogue

d In the *Select Type* dialogue, find the type definition for the attribute in the imported class in your model, then click OK.



e In the *Attributes* dialogue, click *Save* ((see (3) in Figure 4).

f Repeat for all other attributes in the imported class.

Unfortunately, EA does not indicate that the attribute types now point to ISO types (defined in another model), so you will need to keep track of the attributes you have mapped. Run the Conformance Checks tool (see p 15) to highlight any mappings you may have missed.

3.2.4 POSTCONDITIONS

The EA project file contains the information model represented in the selected UML profile (in this case, ISO 19100).

3.2.5 POINTS TO NOTE

The imported model will probably contain redundant attributes and implicit relationships that must be modelled explicitly, to define their behaviour. A modeller must then refactor the model to make it conceptually coherent and reusable. Solid Ground provides the System Metadata tool (see page 26) to assist with extracting common attributes.

In the future, OWL models may be supported by Solid Ground. However, initial experiments have highlighted that individual OWL authors have widely-varying approaches to handling the basics of how features interrelate. It is difficult to predict how the concepts in ISO 19109 General Feature Model are expressed in a given OWL model.

Solid Ground translates database fields to UML constructs using XML-based mapping files. See:

C:\Program Files (x86)\CSIRO\MDA_Utilities\Solid Ground\mappings

for a list of files. These files may edited to suit specific modelling requirements.

3.3 Refactoring existing models using the *Remove System Metadata* tool

Menu: Extensions > Solid Ground > System Metadata.

3.3.1 RATIONALE

The System Metadata tool is a new tool for the Solid Ground June 2012 release. It assists with extracting common attributes from multiple classes, and placing these common attributes into a new, parent class. This process is called *refactoring*.

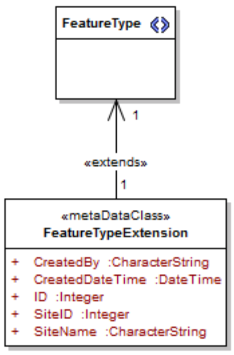
The common attributes in the parent can be regarded as *metadata* for the original classes. In this context, *system* means the classes for which you have extracted metadata, not necessary the entire set of classes comprising the model.

3.3.2 PRECONDITIONS

This tool operates on classes stereotyped with <<FeatureType>>. If you have imported a model from an existing SQL database or from an ArcGIS workspace, ensure that each class has this stereotype.

3.3.3 INSTRUCTIONS

1. In the EA project browser, right-click the package for which you wish to generate metadata.
2. Select *Extensions > Solid Ground > System Metadata > Generate System Metadata Model*.
3. Solid Ground creates a new diagram called *SystemMetadata*. It contains a class, *FeatureTypeExtension*, stereotyped with <<FeatureTypeExtension>>, that extends the base class *FeatureType*. Using the example from ##scenario 1.3, the diagram appears as follows:



What IS FeatureType<<>> here?

The user should rename the class after extracting metadata, right?

Do the metadata classes & diagrams need to be kept with the model?

3.3.4 POSTCONDITIONS

The selected Appication Schemano longer contains attributes designated as implementation metadata by the selected metadata model.

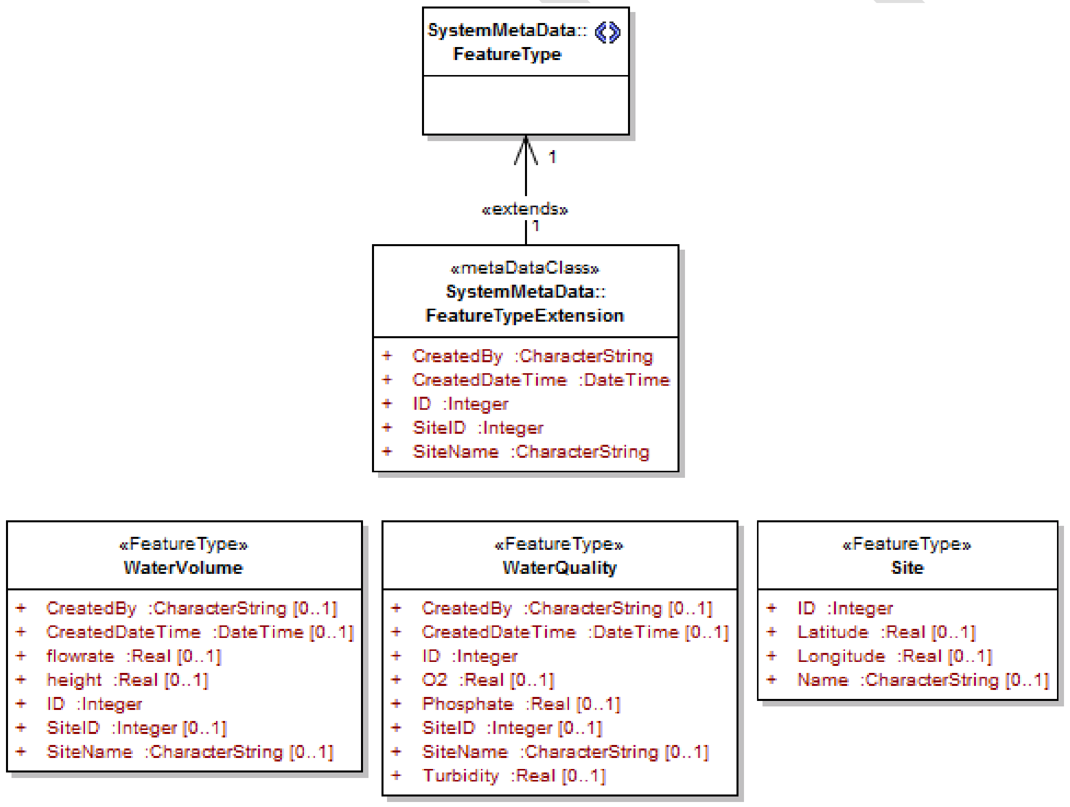
(Add performs the inverse function and adds such attributes)

3.3.5 POINTS TO NOTE

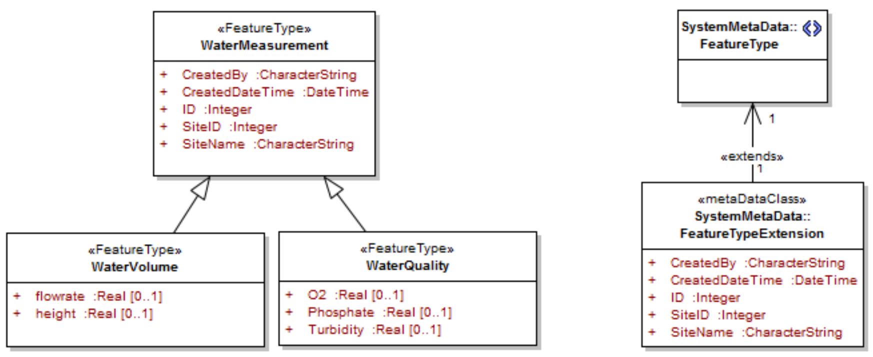
The System Metadata tool extracts attributes common across all classes within the package selected in the EA Project Browser. This means that if you select a package containing Classes 1, 2 and 3, each having a common attribute A, and also classes 2 and 3 have a common attribute B, then the tool will only consider attribute A as metadata – it is the only attribute common across all selected classes (ie across the *system*).

In a complex system, you can refactor similar classes manually as follows:

1. Create a temporary package.
2. Move the classes from their original package into the temporary package.
3. Right-click the package and *Extensions > Solid Ground > System Metadata > Generate System Metadata Model.*
4. ##Needs rewriting## Create the common "metadata" classes and delete common attributes from the other classes.
5. Move the classes back to their original packages, or to other packages if required.
6. Using the example above, you can put the metadata classes on the same diagram as the original classes, making it easy to see what attributes need to be factored out:



7. It's now easy to see which attributes are common. Once these have been factored out, the class diagram should appear as follows:



The classes “WaterVolume” and “WaterQuality” have common attributes CreatedBy and CreateDateTime, ID, SiteID and SiteName. The classes can be refactored to produce a single parent class that contains the common attributes.

The parent class needs to be stereotyped with <<FeatureType>>.

8. The classes can be refactored further, eg removing the attribute SiteName from *WaterMeasurement*, but this is not shown here. Also, for this example, the *Site* class will be regarded as a separate model.

Using a common modelling language, such as using the ISO 10103 Conceptual Schema Language to describe the logic of a formal model, provides predictability in a model-driven architecture, stylistic consistency and avoids of ambiguous modelling constructs.

You can use the "Hollow World" tool to assist creating a clean and conformant Application Schema. For more details see: https://www.seegrid.csiro.au/twiki/bin/view/AppSchemas/HollowWorldHelperPlugin. (Atkinson, 2009)

3.4 Generate Platform-Specific Model (UML to database)

3.4.1 RATIONALE

This tool provides a configurable ability to transform an information model - ie a Platform Independent Model (PIM) to a Platform Specific Model (PSM), or database schema.

Most UML tools provide some UML-to-database capability, but Solid Ground specifically supports the concept of a *shared domain model*, comprising model libraries or reusable model components.

Solid Ground's UML-to-PSM tool can create a database schema for each package in a multipart model, allowing isolation of changes when the model is used to document and maintain complex databases.

In addition, the tool allows complex data types (such a spatial and temporal types) to be mapped to multiple database columns. Tables are created for each Feature Type class. Inheritance is handled by extension tables and views to implement joins.

3.4.2 PRECONDITIONS

The information model exists in a standalone EA project file or in a shared modelling repository.

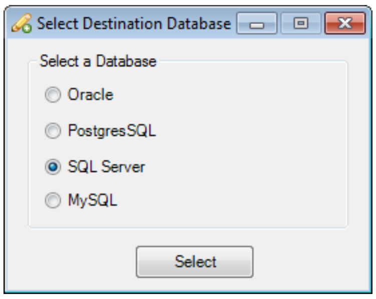
The information model exists in a single hierarchy within the EA project browser.

The information model optionally, but usually, has dependencies on other models. These may exist in other hierarchies within the EA project file, or shared modelling repository.

The UML representation of the model is well-formed, and/or conforms to a particular UML profile.

3.4.3 INSTRUCTIONS

1. In the EA Project Browser, right-click the package containing the model.
2. From the pop-up menu, select *Extensions > Solid Ground > Generate PSM > UML to Database*.
3. Select the destination database type. When the UML-to-database tool generates SQL Data Definition Language (DDL) for the UML model, it conforms to the database type that you select here.

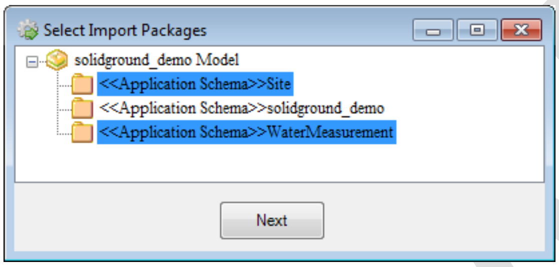


Select Destination Database

Select a Database

( ) Oracle
( ) PostgresSQL
[✓ SQL Server]
( ) MySQL

4. From the *Select Import Packages* dialogue, control-click to select one or more packages to transform to a database schema.



Select Import Packages

solidground_demo Model
<<Application Schema>>Site
<<Application Schema>>solidground_demo
<<Application Schema>>WaterMeasurement

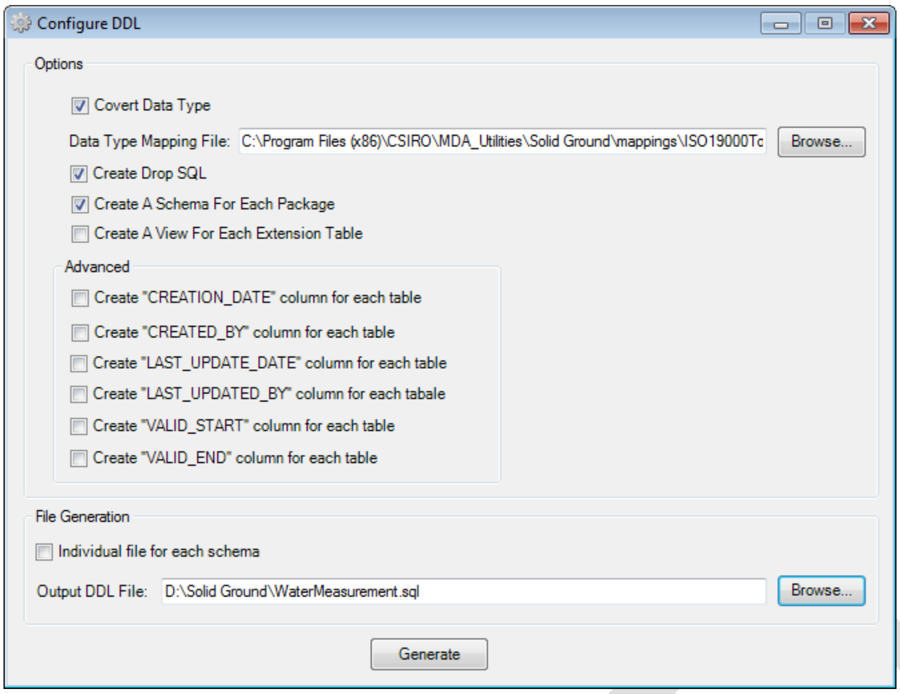
5. Customise the data type mapping file using the *Configure DLL* dialogue. The data type mapping file is a XML file and tells the UML-to-database tool how to map a UML data type to the selected database data type.

   You can customise the mapping file if you wish. For example, to map the ISO 19100 data type "CharacterString" to Oracle data type "VARCHAR2", the following xml element need to be added in the mapping file

   ## give an example here.

   You can use your own customised data type mapping file.

   * in future, each model library can provide mapping files for the packages contained in the library, or one per package #### is this still relevant?


Configure DDL

Options

[✓] Covert Data Type

Data Type Mapping File: C:\Program Files (x86)\CSIRO\MDA_Utilities\Solid Ground\mappings\ISO19000Tc

[✓] Create Drop SQL

[✓] Create A Schema For Each Package

[ ] Create A View For Each Extension Table

Advanced

[ ] Create "CREATION_DATE" column for each table

[ ] Create "CREATED_BY" column for each table

[ ] Create "LAST_UPDATE_DATE" column for each table

[ ] Create "LAST_UPDATED_BY" column for each tabale

[ ] Create "VALID_START" column for each table

[ ] Create "VALID_END" column for each table

File Generation

[ ] Individual file for each schema

Output DDL File: D:\Solid Ground\WaterMeasurement.sql

What are/where are the extension tables?

What purpose do the views of these tables serve in the model?

What is a covert data type? If this should actually be "convert data type", what does it mean?

What is the purpose of the extra columns?

OPTION	NOTES
Covert data type	
Data type mapping file	Specifies which data type mapping file to use. Solid Ground supplies preconfigured mapping files for four database types. You can customise these files or provide your own.
Create Drop SQL	Drop any existing SQL tables and indexes; this prevents errors if you run the script multiple times (for example, if you are testing database generation). Note: the tool currently does not drop views, packages or schemas; you will need to drop these manually.
Create a schema for each package	This option creates a schema for each package stereotyped with <<Application Schema>>. Remember (see section 2.3.5 on page 16) you must not have nested <<Application Schema>> packages Is this to assist with implementing granular models? Perhaps to help with partitioning on physical/logical hardware?
Create a view for each extension table	Purpose?
Create "CREATION_DATE" column for each table	Purpose?
Create "CREATED_BY" column for each table	
Create "CREATION_DATE" column for each table	
Create "LAST_UPDATE_DATE" column for each table	
Create "LAST_UPDATED_BY" column for each table	
Create "VALID_START" column for each table	

Create "VALID_END" column for each table	
Individual file for each schema	If selected, this option makes the tool output separate files for each schema within the database; the Output DDL file
Output DDL file	

5. Provide the target location and name for the generated DDL file.

3.4.4 POSTCONDITIONS

The UML-to-database tool creates one or more .sql script files in the location specified in the Configure DDL dialogue.

3.4.5 POINTS TO REMEMBER

The UML-to-database tool tool currently drops tables from the target database, but not views, packages or schemas: you will get "<entity> already defined" errors if you run the .sql script multiple times.

3.5 Generate Data Migration Script

3.5.1 RATIONALE

This tools takes a concept/implementation mapping and generates either a:

- view to create the implementation model from a database that implements the full conceptual model
- a script to insert data into a model from data selected from a mapped model.

3.5.2 PRECONDITIONS

This tool can be run only on a package stereotyped with <<Mapping Package>>.

These scripts are proof-of-concept only - and relate closely to the idea of modelling data products as simplified profiles of a conceptual model, and using the conceptual model to create a data warehouse capable of storing any modelled data.