Using the SID

Created by Unknown User (joe.fernandez@optus.com.au)

Using the SID

One of the most frequently asked questions about the SID is:

“How is the SID meant to be used?”

Broadly, what people usually mean is:
* Is the SID meant to be used only as a Reference Model, to provide general guidance on how to structure all the data with which an organization deals

or

* Is the SID capable of being adopted as a Canonical Data Model used to implement new interfaces to existing systems, and to implement new applications with new interfaces?

Well, a common mantra within the TMForum (and within the Information Framework community in particular – no names here) is “It’s all about choice”. So an organization can choose either of the above two options.

However, this nugget focuses on the advantages of picking the second choice and offers guidelines on how it can be done.

The SID represents a comprehensive high level model of information that an organization will use for its business processes. Consequently, if an organization adopts it as a Canonical Data Model, the organization saves the considerable amount of time and effort that it would otherwise expend in creating this from scratch.

The specifications are available in text form and UML export format and in XSDs, both Strong Typed and Weak Typed versions. This gives an organization a choice of many tools that can be used to work with its adopted Canonical Data Model, either at the level of graphical modelling using UML diagrams, or at the level of the logical schema definitions (XSDs).

Let’s consider a hypothetical example.

Acme Telco has developed a Trouble Ticketing system which is used by its staff to record all problems reported its customers and by its network operations personnel, and to track the process of problem resolution, and progress with problem resolution from initial reporting to closure.

Acme Telco now wants to provide an interface to this existing system that will allow it to be integrated with new systems being developed, both in house and by external parties.

The interface will handle operations on a fundamental data type, the Trouble Ticket.

Without getting into the details of modelling a Trouble Ticket, we can see at a high level that it will most likely be comprised of information on:
* Affected Service(s)
* Resource fault(s)
* Affected Customer(s) (names and contact details)
* Times (problem detection, resolution ,etc)
* State changes
* Operations personnel (engaged in fault reporting, tracking, repair, etc).

Acme Telco finds that the SID has ABEs to model:
* Trouble Ticket
* Customer
* Resource
* Service
* Base Types such as Time Periods

The Acme Telco teams developing the other applications that will interconnect with the Trouble Ticketing system, and with each other, come to similar conclusions.

Acme Telco decides that it can save considerable time and effort by using the SID as a basis for the data model for its new interfaces and applications. At this stage, both the options listed at the start of this article are available to Acme Telco i.e. use the SID only as a Reference Model or use it also as the basis for implementation.

Looking at the above SID ABEs, Acme Telco finds they do not have the attributes to store all the information that Acme Telco considers essential for creating a Trouble Ticket, or all the information that needs to be displayed to customers or other parties making enquires on Trouble Tickets (all of these being functions that will be performed using the new applications or new interfaces).

To use the SID for implementation, Acme Telco therefore would need to extend the SID model.

There are two general techniques for extending the SID:
* The traditional specialization or subclassing used in object oriented modelling
* Use of the Specification or Characteristic patterns (described in another Nugget)

Acme Telco reasons that the basic structure of a Trouble Ticket does not change very often.

Acme Telco also reasons that subclassing ensures strong type checking and will lead to much greater reliability of the generated code.

Acme Telco therefore decides to create new subclasses of the SID classes. For example:
* AcmeTroubleTicket
* AcmeCustomer

Acme Telco creates a simple document “Acme Trouble Ticket Data Model”, using the TMForum SID Addendum template, to document the subclasses (which are specialized from the SID classes) in the same format. This makes it easy for the reading audience to understand the extended model and its basis on the SID.

One issue that immediately comes up is the names of entities and attributes. Acme Telco adopts the rule that all SID entities and attributes should retain the names in the SID specifications for consistency in identification and for “across the wire” interoperability. For example, SID Trouble Ticket is a subclass of SID Business Interaction and therefore inherits the attribute interactionDate. This is documented in “Acme Trouble Ticket Data Model” to be the Trouble Ticket creation date but its name is retained as interactionDate. New classes that are defined in the extensions have prefixes to their class names (e. g. “Acme”).

Acme Telco considers using a modelling tool to work with the SID UML specifications to extend the model but decides for its initial implementation to use the XML Schema Definitions.

As noted above, SID XML Schema Definitions are issued in two forms, Strong Typed and Weak Typed. In the former, the data types of all elements are declared, whereas in the latter they are not. The Acme Telco team dealing with the Trouble Ticket interface favours strong typing as an approach because it leads to explicit identification of dependencies on referenced data types and allows rigorous type checking, that will, in turn, generate more reliable code.

The SID model has a very large number of entity definitions. For performance reasons, Acme Telco does not wish to include the large number of unused entity definitions in the data model that is being implemented. Hence it wishes to sub-set the SID data model (as well as extending it as described above). This is best performed by using the strong typed version of the schema definitions, which allows the designer to select the required base classes, identify the dependencies on other classes, include these but exclude or selectively disable elements that will not be used.

The designer starts with the SID file SIDPhaseIXModelsStrongTyping.xsd which contains all the SID definitions in strong typed form. He uses the XML editor in Eclipse to select a subset of this file holding just the type definitions of the SID entities to be used by the interface and where appropriate, comments out association elements that are unused and that refer to entities that are not being selected.

The designer then creates a second file AcmeTroubleTicket.xsd that holds the subclass element definitions as extensions of the SID base classes. Importing the first file allows the model syntax to be validated.

Finally the designer creates a minimal WSDL file AcmeTroubleTicket.wsdl that specifies the required operations (Create, Get, Set) on the Acme Telco data types defined in the second file. The bulk of the specification has been done in the AcmeTroubleTicket.xsd file.

These three files, together with the Data Model document, represent a complete specification of the new SID - conformant interface.

In summary, in this exercise, the Acme Telco team has:
* adopted the SID as a Canonical Data Model
* identified the relevant entities that will be the super classes in the Canonical Data Model
* modelled the subclasses to reflect Acme Telco’s data
* edited the SID 9 Strong Typed XSD down to a subset consisting of the chosen super classes type definitions
* created an XSD of the modelled subclass type definitions
* added a thin WSDL layer.

This will now allow the use of tools to generate a Web Services interface from the WSDL.