Resource Modeling Sub-Project deliverables

In most tools that monitor resources there is an awareness of some of the more static features of a resource; for example the physical location and the IP address, in addition to the actual data being extracted from the resource is often known. Yet each tool has its own representation of this information. With the growing presence of distributed and componentized applications this information also includes the overall topology of the application. This is redundant information between the tool used to debug a performance bottleneck and the tool that is used to deploy components of the application, or the tool that is doing load testing and monitoring of the application. The Resource Modeling sub project goal is to address this redundancy by helping to build a canonical representation of these resources.

The COSMOS Resource Modeling sub-project aims to provide support for building a common model to represent the information being shared in a system management scenario. The project is proposing to use SML as the xml schema language to define this common model. The SML-IF interchange format will be used to share model components between the tools involved in a system management activity.

To that extend, the COSMOS Resource Modeling sub-project is focused on the following deliverables:

1. Tooling for validating SML instances and SML-IF documents
2. Tooling for importing/exporting SML-IF documents to and from predefined repositories.
3. Tooling for creating SML template documents
4. Tooling for creating domain models based on existing SML templates

The Resource Modeling workgroup is proposing to investigate which Eclipse projects are offering XML editors or schema validation and work together to create a common scenario to make Cosmos and these projects interact and complement each other.

WTP has been identified as one of the projects offering xml schema and instance validation. The action is available for any document with the extension XML and XSD (right click and select Run Validate or Validation XML File action). A good usability scenario would require extending these actions with SML extra validation and not providing new validate actions on an xml or xsd resource. Since this may require working with the WTP team to define a common scenario and the set of extensions required to hook in SML validation, this may not be a viable delivery for the first release of Cosmos. This should not stop us from starting the collaboration with WTP (and any other projects) with the goal of identifying required API and extension points for making this happen.

Detailed description of the above deliverables:

Tooling for validating SML instances and SML-IF documents

Target date for this item: March 07.

The validator should be fully functional by the end of the COSMOS 1.0 , June 07

1. Provide a common interface for handling and reporting errors
2. SML validation
    a. Provide a common interface for handling and reporting errors
    b. A default implementation for the XML validation phase
    c. A default implementation for the SML extension validation phase
    d. A default implementation for the Schematron validation phase, which will include displaying results to a predefined output based on report/assertion conditions
    e. Create a default validation scheme for the SML extensions: sml:acyclic, sml:targetElement, sml:targetType, sml:key, sml:unique, and sml:keyref
3. SML-IF validation
    a. A default implementation for the XML validation phase
    b. A default implementation for the Schematron validation phase in the context of an SML-IF document; includes displaying results to a predefined output based on report/assertion conditions
4. Apply the SML validation extension in the context of an SML-IF document: sml:acyclic, sml:targetElement, sml:targetType, sml:key, sml:unique, and sml:keyref
5. An implementation of RFC 3986 to resolve inter-document references within an SML-IF document
6. Verify that the SML and SML-IF validation implementations cover the test cases of the SML workshop scenario

Tooling for importing/exporting SML-IF documents to and from predefined repositories.

Target date for this item : March 07

An SML-IF document can be formed from a project by selecting export from the context menu of one or more resource that appears under a project. The operation will determine the genic or phenic documents based on a registered content type that describes each type of the document. The user will have the option of changing the document type (i.e. genic/phenic) if a unit's type is incorrectly assigned. The export operation will be integrated with Eclipse's export wizard under the category: "Service Model Language".

The user can also import an SML-IF model into its units using the import operation. The import operation will simply divide up an SML-IF document into individual genic and phenic documents and store them in a desired target folder. The import operation will be integrated with Eclipse's import wizard under the category: "Service Model Language".

Ideally an import operation followed by an export operation should generate a file that is semantically equivalent to the original file imported. Unfortunately there is some loss of data when an SML-IF document is broken into its respective SML model units. For example, the aliases associated with a document or schematron rule bindings are not preserved in SML model units. To preserve this information, a meta-data file is generated with every SML-IF document import. Using the meta-data, the user can perform an export operation that will result in a file that is semantically equivalent to the original SML-IF document imported. Validation can be done at a project, folder, or a unit level. To validate an entire model the user can select validate from the context menu of a project. Alternatively the user can validate a subset of the model by selecting validate from the context menu of a parent folder or a set of model units.

A more detailed document describing import and export implementation and extension points is attached below.
http://wiki.eclipse.org/images/a/ae/SML-Validation-v7.pdf

Tooling for creating SML template documents

Target date for this item: post June 07

An SML template document is an SML instance defining a common pattern that can be re-used and adapted in different domain models.
The project is proposing to offer an extendable tool which allows users to easily create SML templates. The tool will be based on GEF and will provide drag and drop capabilities for creating templates.

Tooling for creating domain models based on existing SML templates

Target date for this item: post June 07

The project is proposing to offer an extendable tool which allows users to easily create SML instances based on SML templates. The tool will be based on GEF and will provide

drag and drop capabilities for building the instances. The SML instances will be possibly created using SML templates, built using the SML templates builder.

The tool should be extendable to allow registration of third parties SML templates.