Primary use cases of MBSE libraries

The INCOSE MBSE Usability Group voted on the most important systems usability issues to investigate at the January 2011 working group. The working group selected the library use case. This paper begins by understanding the value of a brick and mortar library. Using the library analogy, use cases are developed to describe basic concepts needed to use, develop, and maintain the library elements. Finally, usability issues (**UI**) are identified. Usability issues reduce user efficiencies, increase the chance of user errors, decrease the understandability, or increase the time it takes to learn. The desire of this paper is encourage a discussion leading to more usable languages and tools.

To understand MBSE libraries, it is helpful to think of the original definition of a library. Webster's Student Dictionary of 1938 defines a library as a room or building given over to a collection of books kept for use and not for sale; also an institution for managing such a collection. Webster's definition of a library can be extended to the MBSE library. A MBSE library is as a set of language, process, and tools needed for an enterprise to create, maintain, organize and use a collection of common foundational model library elements. Like a book, a library element is a self-contained publication that can be referenced and re-used to create new model elements.

To use and manage MBSE libraries, there are three basic actors, the Product Modeler Actor, the Product Support Actor, and the Library Modeler Actor. See Figure 1 below.

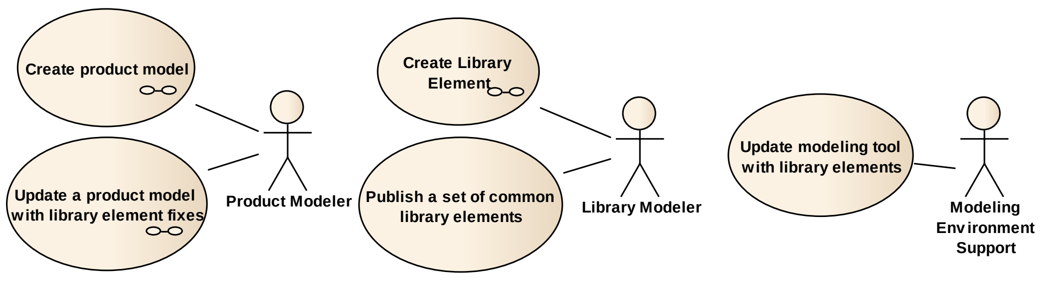
Create product model
Update a product model with library element fixes
Product Modeler
Create Library Element
Publish a set of common library elements
Library Modeler
Update modeling tool with library elements
Modeling Environment Support

Figure 1: MBSE Library Use Cases

This section of the paper focuses on one of these five use cases, the Create Product Model use case. The Product Modeler creates a product model, a behavior or structure model of the product, to meet a modeling objective. The modeling objective includes both the understanding of what the model needs to do and an understanding of the set of results. Example modeling objectives may include diverse objectives such as modeling a software behavior used to generating target software, modeling a network used to determine the correct number of switches, or modeling system behaviors to elicit customer needs. To illustrate this use case, a very simple example of a low-pass filter testing model is developed. The objective of this example model is to verify that the rise time of the low-pass is consistent with the time constant. The example model is built using a SysML Activity Diagram.

The product modeler creates a product model by following the actions for the use case shown in Figure 2.

The **pre-condition** for this use case is a list of library elements that is available and ready to use. The actions for this use case are listed below:

1. **Capture what the model needs to do (Requirements, use case, and/or test)** – Product models are built for a purpose. The requirements, use cases, and/or tests capture this purpose. The product modelers require the ability to change the value of Tau for each instance of the Low pass filter. The tester must verify that Tau can be changed.
2. **Find library element** - In this example, the Step Function, low pass, and Time History Plot library elements are found in the list of all library elements. The library elements are implemented as SysML activities. The activities contain all of the behaviors needed to execute the model.
    a. Note: Users will often use 20% of the library elements for 80% of the model.
    b. **UI 1** – It is hard to find, identify, and understand the usage of library elements in a model. Tools must minimize the time it takes to find and understand what the library element is and how to use it. Understanding comes from both an example and a description that includes the environmental constraints the library element requirements, and the realized tests.

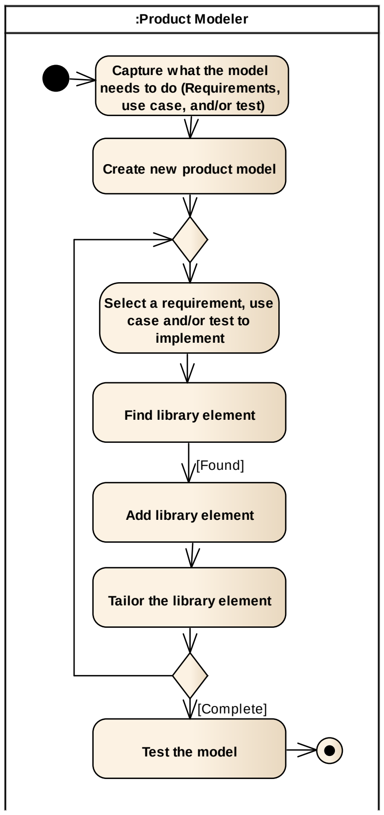
:Product Modeler
Capture what the model needs to do (Requirements, use case, and/or test)
Create new product model
Select a requirement, use case and/or test to implement
Find library element
[Found]
Add library element
Tailor the library element
[Complete]
Test the model

Figure 2: Actions for Create Product Model Use Case

3. **Add library element** - Create an instance of the library element in the editor. In this example, a step function action is placed in the editor. It becomes a call action. The step function has one output and no inputs. The pin placement is visible and is located in the same place as the parameter on the library element activity. The icon draws a picture of a step function and scales the picture to mirror the default value of 1 for the step input and 1 for the step size. See the diagram below:
    a. **UI 2** – It takes a number of steps to re-size the action and to place the pins around the perimeter of the element. Tool vendors need to minimize the number of steps needed to re-size and adjust the location of the pins. As an example of how to do this, some vendors have used a template layout to position the pins and to size the action.
    b. **UI 3** – There is no graphical way to see which pin is an input and which pin is an output.
    c. **UI 4** – There is no language support for developing custom icons.

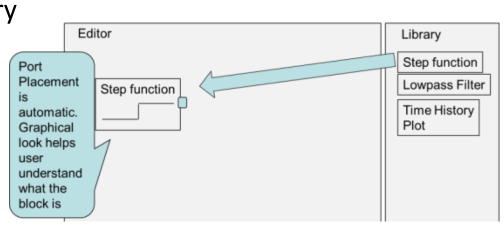
Editor
Library
Port Placement is automatic. Graphical look helps user understand what the block is
Step function
Step function
Lowpass Filter
Time History Plot

Figure 3: Add Library Element

4. **Tailor the library element** – Set the variant of a library element to a value needed by the product model. A variant is a library element parameter that can change the form or function of the library element. Tailoring is the process needed to customize the library element for a specific use in the product model. In this example we tailor the function instance by setting the step insertion time at 1 second and the step size at a value of 1.
   a. **UI 5 –** It is difficult to look at a diagram containing a library element and see the items that can be tailored on a library element and to see the value chosen for a tailoring.
   b. **UI 6** – It is easy to introduce errors in a model when pins are connected together with an object flow. This is because the pins connected together with the object flow may have incompatible types.

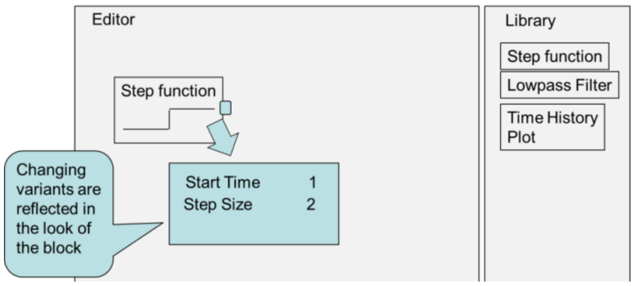

Figure 4: Tailor the Library Element

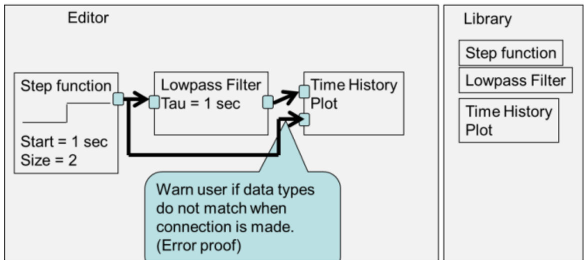

Figure 5: Error Proof

5. **Test the model** – Verifying the rise time of the low-pass filter involves executing the model, looking at the time-history plot, and measuring the rise time.
   a. **UI 7** – It is difficult to execute models. Sometimes executing models involves creating classes and placing the activities below the class, and writing software. This process should be easy to setup and easy to understand.
   b. **UI 8** – It is difficult to visualize the results of an executable model. Capabilities such as the ability to single step, watch the value of a pin, and plot the time-history of a pin help understand the dynamics of a simulation and aid in testing.

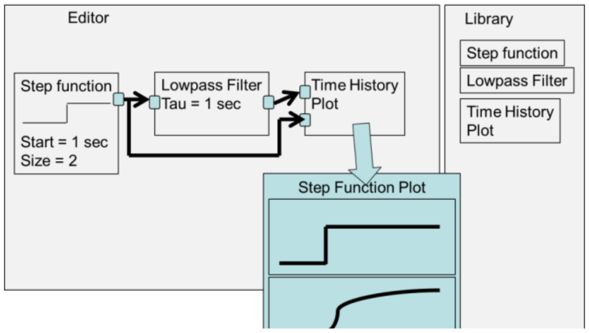

Figure 6: Plot Simulation Results

Each of the remaining use cases un-cover additional usability issues. This is a subject for a future paper. A summary of the use cases is described in the remaining part of this section of the paper.

Update a product model with library element fixes

Library elements will change over time. Changes may be driven by events such as feature enhancements, dependencies, or bug fixes. The product modeler needs a way to update the product model with changes. It is possible that changes to the library element will require fixes to the product model that use the library element. A product modeler may also choose to not accept changes to a library element. This may happen if the change is an enhancement to the library element that is not needed by the product.

UI 9 – It is difficult to identify the dependencies on a library element change. Dependencies may include new model elements that need to be loaded into the model because of an update and assumptions on the environment that the model will work in.

Publish a set of common library elements

Common library elements only have value in an organization if they can be found and maintained over time. The Library Modeler publishes common library elements and notifies the product team (product support and product modelers).

UI 10 – It is difficult to package and share multiple elements in a sharable library element. Elements include requirements, the design, the implementation, the tests, and the test results.

UI 11 – It is difficult to document the help on a library element.

Create library element

Creating library elements starts with creating a product model. Some of the model elements used to create the product model is usable across multiple products in a product line, across all models in an enterprise or across all models in a modeling discipline. The Library Modeler considers the requirements of the user group and updates the library element, documents the library element, and tests the element to show that it meets the requirements.

Update modeling tool with library elements

Finally, the Modeling Environment Support person updates the modeling tool when new or changed library elements are available. If desired, the product team also needs a way to update the existing models to use the new and improved library elements. Product teams want the option to update to the new library elements. To make this decision, the product teams need to understand the differences between the version of the library element they are using and the new published version. They also want the option to either upgrade or to not upgrade to new library changes.

UI 12 - It is difficult to configuration manage multiple versions of a library element in a model. Managing includes capabilities such as selecting only one version of a library element for use in a model, upgrading to a new version of a library element, and differencing one version of a library element with another.

Conclusion

Creating product models from library elements is the most important use case selected by the MBSE Usability Group. This paper details the use case steps needed to build and manage a model from library elements. The roles of a product modeler, product supporter, and library modeler are described. The product modelers use the modeling environment to create product models and to update a product model with library element fixes. The library modeler uses the modeling environment to publish a set of common library elements, to create library elements, and to modify library elements. The Product Supporter uses the modeling environment to update the modeling tool with library elements and to find library elements.

The use cases revealed a number of necessary features. For example, there needs to be a way to tailor library elements, to capture environmental dependencies, to capture dependencies on other model elements, and to package numerous artifacts together (requirements, tests, test results, etc.). These features need to be supported by the tools, the tool environments, or by the underlying languages the tools are based upon.