Elvior’s Input on TDL 3 Discussion

# Priorities of TDL concepts for diagram editor

In addition to diagrams, there are 2 kinds of visual editors commonly used for modelling in Eclipse: tree and property sheet. In the absence of full TDL graphical syntax support, such approaches could be used as alternatives. There are concepts in TDL for which the readability can only be achieved by standard graphical notation, like behavior, test configuration. Those concepts should take priority when developing diagram editor.

Concepts in order of priority:

1. Test description behavior
2. Test configuration
3. Definitions with composite symbols
	1. Structured (and simple) type
	2. Simple and structured data instance
	3. Resource and element mapping
	4. Component type
	5. Action and function definition
4. Definitions with simple symbols
	1. Annotation type
	2. Time
	3. Gate type
5. Package
6. Composite forms
	1. Test Description (behavior)

Problematic:

* Multicast interaction
* Variable labels under arrowhead

# Use cases for TDL editor

1. Create model elements, edit attributes and references, delete
2. Create diagrams
3. Drag and drop elements, reconnect connectors
4. Validate model
5. Layout diagrams
6. Diff and merge model

# UML TDL profile vs TDL meta-model for editing

Perhaps more important than the choice of diagramming platform is the choice of underlying meta-model used to base the editor on. Below are listed some arguments for and against using UML TDL profile instead of TDL meta-model. Some of the arguments are specific to chosen diagramming framework and most are possible to work around, but with additional effort.

Drawbacks of using UML profile:

* UML TDL profile not available in the beginning of STF
* UML TDL profile will be more complex (contain more elements)
* Model objects with applied stereotypes can’t be treated as instances of those stereotypes in EMF and related query language implementations such as MDT/OCL and Acceleo
* Reflection based UI controls will show attributes from UML meta-classes not relevant for TDL
* Model validation potentially garbled by UML constraints not relevant for TDL

Benefits of UML profile:

* Model can be read by all UML tools