STIL P1450.4 Rosetta Stone (02/28/06)

Conceptual Model Rev K, 9/28/05	Syntax Document Rev D16, 12/05/05	Reconciled Terminology	Comments
TestProgram	TestProgram	TestProgram	
EntryPoint	EntryPoints	EntryPoints	
TestFlow	TestFlow	TestFlow	
FlowNode	FlowNode	FlowNode	
TestMethod	TestModule	TestMethod	See Note 1
TestBase (sec. 2.4.1)		TestBase	See Note 1
Defaults			See Note 2
	TestMethodDefs	TestFunctionDefs	
TestObject	TestInstances	TestInstances	
TaskNode (sec. 2.7)			Not needed
DecisonNode (sec. 2.7)			Not needed
ObjectRef/ModuleRef	TestExec	TestExec	
PreActions	PreActions	PreActions	
PostActions	PostActions	PostActions	
SkipPath	Bypass	Bypass	See Note 3
Arbiter	*boolean_expr* of FlowNode ExitPort block		
PassActions/FailActions Blocks	ExitPorts	PassActions/FailActions Blocks	See Note 4
ExitActions Block	ExitPorts	ExitPorts	See Note 5
	ReturnState		See Note 6
BinMap	BinMap		
BinNode	BinDefs/Pass/Fail/Bin		

Note 1: TestBase provides the common minimal template for all TestMethods and TestFlows. Its contents have not yet been defined completely, but at the least, it includes the following:

- Parameters common to all TestMethods and TestFlows.
- Internal data which all TestMethods and TestFlows must contain
- Functions which all TestMethods and TestFlows must provide

One question which remains a point of contention is this: Should a TestMethod include a predefined Execute function (which does not need to be stated in the program text, and which implicitly appears between the pre-actions and the post-actions)? Or should the TestMethod include an explicit TestExec statement which specifies what function (in an external library) to execute for this method?

Note 2: Defaults. The syntax, semantics, and content of the defaults block have not yet been defined. The intent, however, is to provide a base set of pre-actions, common post-actions, arbiter, and arbiter-path-specific post-actions, so that a TestMethod, FlowNode, or TestFlow, if it uses these defaults, does NOT need to restate them. It provides a form of shorthand, reducing the verbosity of the program source text – at the expense of local clarity (i.e., if the above elements are NOT included in, say, a FlowNode, then the user must look elsewhere to the defaults to determine what they are set to).

Note 3: There are two forms to the Bypass – one which bypasses to the INPUT of one of the one or more ExitPorts blocks (allowing execution of any actions listed there), and one which bypasses to the OUTPUT of one of those same blocks (skipping execution of any actions there). We believe that we can support both – the former would use the Bypass ReturnState <return_state> syntax, and the latter would use the Bypass GoTo <PORT_LABEL> syntax. See lines 67-71 of the latest syntax document on the web (D16, dated March 1, 2006)

Note 4: Need to decide if we want a separate keyword and syntax for the outcome-specific actions in TestMethod and TestFlow, or to reuse the FlowNode ExitPort syntax

Note 5: See Note 4.

Note 6: See discussion about ReturnState in Note 3, and semantics of ReturnState on p6 (in the discussion about TestExec – section 19 (19.c) in the latest document.