P1450.4 syntax subgroup meeting minutes – 10/03/05

Attendees: Jim O'Reilly, Dave Dowding, Greg Maston, Tony Taylor, Doug Sprague

Not present:

Agenda:

- Discussion about HW setup statements (Timing, DCLevels, DCSets, Category, Selector), and where they belong.
- Discuss whether to use PatternExec instead of the individual elements of PatternExec.
- Discuss areas of the document currently shown in red (indicating not settled and still under discussion)
- Discuss the contents of pre and post actions for TestNodes and TestFlows, as well as the contents of exit port statements

Summary (areas of discussion):

- Discuss conceptual model (particularly section 2.4, which discusses the TestBase->TestMethod (including Flow as a type of TestMethod) inheritance hierarchy.
    - Tony: The description is too specific – it implies a particular implementation. What's described here IS the definition of one implementation, but other interpretations or implementaions are possible. We shouldn't REQUIRE that a STIL .4 system be implemented this way. As a description of ONE possible implementation, this should be moved to an annex.
- Document review: Discussion about pending areas (red-text statements/blocks) in current D15 draft (dated Sept. 26, 2005)
    - Use of PatternExec
        - Unanimous agreement about using the existing PatternExec block to specify HW setup statements (Timing, DCLevels, DCSets, PatternBurst, Category, Selector) – instead of specifying any or all of those elements separately.
    - In which blocks (TestProgram? TestFlow? TestNode?) should we include the PatternExec?
        - Unanimous agreement to drop PatternExec from TestProgram. Continue to include it in both TestFlow and TestNode.
    - TestInstances: Do we want to include TestInstances in both TestProgram and TestFlow?
        - Consensus – yes, it should be included in both places.
    - Include both Category and Selector in *stil_block_type* (lines 1-15)?
        - Consensus: yes
    - Inherit: Do we feel that inheritance would be useful? If so, what should the behavior of derived blocks be, with respect to the elements of the parent or base block?
        - General consensus is that inheritance can be useful – but it should not be required.
        - For Parameter and Variable blocks, the entries specified in the derived block should EXTEND (add to) those specified in the base block.
        - For other elements of the TestNode block (PreActions; PatternExec; TestExec; PostActions; *exit_port* stmt), the entries specified in the derived block COMPLETELY REPLACE those specified in the base block. If you want to accept the set specified in the base block, simply do not specify that element in the derived block. On the other hand - at least for pre- and post-actions – it is possible to ELIMINATE COMPLETELY the pre- and post-actions specified in the base block simply by including the PreAction or PostAction statement with an empty set of actions.
        - See writeup in section 2.11 of latest syntax document (D15, dated 10/03/05)
    - *Instance_stmt*: Include this?
        - Consensus – yes. Semantics are as follows:

- If using the first form (on line 56 – i.e., FUNC_NAME INSTANCE_NAME), no initial values for parameters are supplied at instance creation time, and initial values for ALL parameters MUST be supplied in the definition of that type.
- If using the second form (on lines 57-60), then initial values for some or all parameters can be supplied when the instance is created. In this case, it's possible to have a type definition which does not supply initial values for all parameters.

- *Execute_stmt*: Some discussion about the two different forms – shown on line 50 and on lines 51-54.
  - PROPOSAL – STILL UNDER DISCUSSION:
    - The first form (in which no parameters are specified) can refer either to a type (TestFlow, TestNode, or TestMethod) or an instance of any of those types. Note that there has been NO change in semantics for this first form.
      - If it refers to a type, an anonymous (inline) instance is created – but since no initial values for parameters are supplied at time of instance creation, ALL parameters MUST have initial values provided in the type definition.
      - If, on the other hand, it refers to an instance, that instance (which had initial values for all parameters provided either by the type definition OR when the instance was created), is called.
      - There has been no change in our sema
    - The second form, which specifies initial values for some (or all) parameters, can refer ONLY to a type, and not an instance. It's this second form for which there has been a change in semantics. Previously, we were allowing (intentionally or not) the initial values of parameters for an instance to be set
      - At type definition type (for the type – still allowed)
      - At instance creation time (for the instance – still allowed)
      - At instance execution time.
    - It's the third item above that we're currently discussing – do we want to allow initial values for parameters of an instance to be superceded by values passed at call time? The jury is still out. If we accept the proposal, then initial values for parameters of an instance can be specified ONLY by the type definition or when the instance is created, but NOT when the instance is executed.

For reference STIL .4 information can be found at the IEEE STIL website:
http://grouper.ieee.org/groups/1450/ (select the P1450.4 link from the table) or use the direct link
http://grouper.ieee.org/groups/1450/dot4/index.html