P1450.4 syntax subgroup meeting minutes - 08/29/05

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

Not present:

Summary (areas of discussion):

- Parameter block vs. many independent parameter statements.
  - Proposal accepted.
- Scoping of variables.
  - Scoping of variables created by a **Variables <var_domain>** keyword within a block is local to that block.
    - In order for any of those locally-scoped variables to be visible to a callee (that is, to a lower-level), they must be passed to the callee as parameters.
    - This is in contrast to the scoping of variables in P1450.1; in P1450.1, any variables created at a specific level are visible to that level AND to any level beneath the level at which they were created. After much discussion, the syntax WG decided that, for P1450.4, the situation regarding variables and scoping/visibility at levels below the the level at which they were created was different enough in circumstance and intent from P1450.1 to warrant different handling.
    - Therefore, the environment for a given block in P1450.4 includes any variables created by one or more **Variables <var_domain>** keywords, AND any variables passed as parameters.
    - This course of action is not really in conflict with the rules of P1450.1; rather, it's an extension. As stated in .1, the source for substitions in since
    - Not really going against .1 – elaborating – when a pattern uses # or % assignment in a macrodef, that variable had to be declared in a parameter block, which doesn't exist in .1
- Order dependence of statements.
  - Proposal – TestExec is a block, rather than a single statement. Keep Pre/Post Actions blocks. Order of execution of PreAction, TestExec, and PostActions is implicitly in that order, regardless of how they're listed in the STIL source file.
    - We had some discussion about whether we wanted to have a single Action statement, which could be placed either before or after the TestExec (possibly having multiple TestExec statements with various Actions intermingled among them). In such a case, whether an Action was Pre or Post would be entirely be determined by where it occurred relative to a TestExec statement.
    - Differentiation of pre and post-actions – is there any difference between pre and post actions? If so, that argues FOR having separate Pre and Post actions blocks. If not, that argues AGAINST having separate Pre and Post actions blocks.
    - In the end, we decided to retain the PreActions and PostActions, since our previous work has shown that there are some differences between the list of actions allowed in PreActions and PostActions. Thus, we retain a single PreActions and a single PostActions block (each of which can contain mutiple (possibly conditional) actions), and a single TestExec.
    - Note that pre and post-actions are mainly intended to allow operation on flow-related variables, rather than general STIL variables such as signal, timing, or DC levels blocks.

- Lengthy discussion about the implications of parameter passing into blocks with respect to data substitution in Procedures and MacroDefs, and WHEN a passed parameter gets resolved.
    - If TestMethod (with passed parameters) is a pattern, it MUST behave based on the 1450.0 and 1450.1 pattern handling behaviors:

      to wit: in a pattern with a procedure MyProc
      ```
      MyProc
      {
          foo=#;  // foo = 2 until it's assigned via a # or % in the body of MyProc
      }
      Foo=2;
      Call MyProc  { foo=3; }
      ```

    - Haven't defined (yet) what TestMethods STIL will include (most have been deferred to .5), but the two Methods we feel should probably be intrinsic to .4 are Functional Test (PatternExec?) and Flow execution (is flow simply a type TestMethod, or something distinct from TestMethod which happens to behave like TestMethod).
- Dave – Chris Nelson asked that we look at how OTPL deals with variables scoping. Chris has since summarized these rules and forwarded them to us. We'll look at these – Tony will write up a document which compares STIL and OTPL, so that we can consider how much commonality we want to have with OTPL.
- TestMethod vs. TestObject. Beginnings of a discussion about swapping the concepts/contents of TestMethod and TestObject. Currently, TestMethod is simply an interface spec, while TestObject adds the pre and post actions. Based on conversations with other WG members, the TestMethod might be (should be?) considered a template, which contains the pre and post actions, while the TestObject should be just be the instantiation (including filling in values for parameters) of a TestMethod. Not resolved yet – stay tuned.

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