P1450.4 syntax subgroup meeting minutes - 08/22/05

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

Not present:

Notes:

- Minutes are available on the website – Click on "syntax-group" in "*Quick Links:*" section near top of main P1450.4 web page (see address below).

Summary (areas of discussion):

- Summarized for syntax WG the issues arising from the full WG meeting on Wed., 08/17/05 (in which we discussed the contents of the most up-to-date syntax draft.
    - TestObject block needs a way to specify TestMethod name.
    - Should Category/Selector be handled separately and differently from other variables; or should they be just like other variables
    - Microflows – need to string together several integral test method to create a user-defined test method. One way to handle that is via a subflow; another is to allow multiple TestMethod calls where (currently) one would be allowed. In this latter case, how does the caller of the multiple TestMethods determine which return state to use? Is it from a
    - Can variables blocks be created inside another block (to provide private variables for that block)?
- Discussion of current state of syntax document. Discussion focussed a couple of changes made by Tony
    - Keyword **TestExec** removed from *execute_stmt* – simplifies and clarifies syntax structure; TestExec keyword is added before execute_stmt in TestObjectDefs and FlowNode (should this be TestNode or TestFlowNode?)
    - Added category, selector, and variables into TestProgram block. If declared in TestProgram, global to entire program; if in TestFlow, then local to that flow; if in a TestObject, then local to that object.
    - Need to change VAR_NAME references in Variables statement to VAR_DOMAIN.
- Discussion about the scoping of variables. What is the scope of variables declared in a variables block, and instantiated in a block via a Variable <var_domain> statement?
    - Greg clarified the rules (as established per P1450.1) – any variables block instantiated in a (parent) block creates a separate instance of those variables, and no other block outside the scope of the (parent) block can access those variables.
    - It's important to note, that unlike C/C++ notions, variables instantiated at an upper level are automatically visible to callees of the block creating those variables – without the need to explicitly pass any or all of those variables as parameters to the callees. We need to think about this, and decide if we want to follow the .1 rules, or if the circumstances which apply for .4 are different enough from the circumstances which apply for .1 which led to that rule to warrant different handling.
- Need to clarify and describe all rules and issues surrounding the operation and visibility of variables at different levels (i.e., are variables established at an upper level automatically visible at lower (callee) levels? What is the benefit of parameter passing if the previous is true?).
    - In other words, if variables are only visible at the level at which they're established (unless passed to a lower level), that constitutes static scoping. Greg: currently, in STIL, Variables blocks are not statically scoped – they're dynamically scoped (that is, instances are created whenever a block which contains a Variables <var_domain> statement 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