# 1450.4 meeting minutes - 10/09/08

**Attendees**: Doug Sprague, Jim O'Reilly, Ernie Wahl, Bruce Parnas, Ajay Khoche

**Not present**:

**Agenda:**

- Preamble:
  - Record Meeting (*2) (call not recorded, since we did not have a quorum)
    - To listen to the meeting recording, do the following:
      - Call the (US) dial-in numbers 1-877-421-0003 (toll free) or 1-770-615-1374 (toll)
      - Enter the passcode code 747464
      - Once dialed in with the proper access code, enter *3 (star 3)
      - Then enter the file number 23119101 for this conference (this number will change each week).
  - Press 1 to listen to the conference
  - IEEE Meeting Preamble (No discussion of proprietary information).

- Finalize ITC meeting.
  - Jim, Bruce, and Ajay to be either at ITC or in Santa Clara during the conference. Only day that works for Jim is Wed 10/29. WG agreed to meet for 3 hours, between 10am and 1pm Pacific time at Advantest.
  - Bruce will reserve a room at Advantest's facility.
  - Doug to set up conference line for the call, using his IBM callin number, for those who can't attend the meeting in person. The line will be open for the entire 3 hours; however, during the final hour, we'll wrap up and summarize – so those not attending in person can call in only during the final hour, if they wish.
- Discuss WG chair assignment.
  - Doug will be moving on at the end of the year, we need to vote in a new chair of the WG
  - Doug nominated Jim to be the new chair of the group, seconded by Bruce
  - Group Unanimously voted Jim to be the new chair
  - Need to get someone to take over doing minutes in the coming weeks
  - Between Bruce or Ajay, should hopefully be able to get a phone line to use
  - Jim would like to get a few other reps for the group, maybe from the usage side of STIL
- Review vars/params changes in latest (D26, 10/7/08) syntax document and "Variables and Parameters Example" (in "Docs for Working Group Meetings" dated 10/07/08).
  - In example of initialized integer array shown in "Variables and Parameters Example" document - Integer myInitializedIntArray[4] = [12,11,10,9;] – semicolon needs to be outside closing brace, rather than inside. The syntax described in the syntax document is correct – it was just represented incorrectly in this example.
  - Group reviewed and agreed to the use of dual formats (using both [ ] and [*array_length*] on the left-hand-side) for dimensioning an initialized array. The statements

    Integer myInitializedIntArray[4] = [12,11,10,9];

    and

    Integer myInitializedIntArray[] = [12,11,10,9];

    both create an initialized integer array of 4 elements. In the second form, the array length is determined by the number of initializing elements listed on the right-hand side of the "=".

    - Array dimensions must be fixed at the time the array is declared, and once set, cannot be resized. This means that the *integer_expr* used to define array dimensions must be either a numerical expression (i.e., 4 or 3+5 or 3*2+7), or if a variable is used to dimension an array, that variable must be an integer constant variable. The intent here is to disallow dynamic resizing of arrays after they are declared and dimensioned.
  - Discussed the keyword **None** as an initializing value for parameters in TestType or FlowType definitions.
    - If a parameter is uninitialized, the user MUST provide a value when instantiating the type.
    - If a parameter is initialized, then the user MAY provide a value when instantiating the type – if the user does provide a value, then that value is used. If the user does NOT provide a value, then the initial value specified in the type definition is used. The keyword None can be used as an initial value.
    - If a parameter has a value of **None** when a Test or Flow is executed, that parameter shall be ignored by the runtime code. This situation will occur when a parameter in a TestType or FlowType definition has an initial value of None (meaning the user does not need to provide a value when instantiating the type), and the user does not provide a value other than **None** when instantiating the type.
    - The keyword **None** is valid only when establishing initial values for parameters in TestTypes and FlowTypes. It cannot be used to establish initial values for variables in Variables blocks.
  - Discussed what the semantics of using uninitialized variables (in variables blocks) should be. Group agreed to require all variables in the Variables block to be initialized. Alternatively, we could simply state that it is illegal to read an uninitialized variable; writing an uninitialized variable would be allowed, and would have the same effect as initializing that variable at the time of declaration.
- Discussed need for multiple *execute_stmts* tokens in a **TestExec** statement in *flownode_stmt* (line 178 of D26, Oct 9, 2008 syntax document). This refers to the

  TestExec { ( *execute_stmt* )* }

  form which occurs in a FlowNode. Since we are now allowing bare TestExecs (with the default FlowNode filling in the remainder of the FlowNode contents) anywhere where a FlowNode statement is allowed, the multiple *execute_stmt*s in a TestExec { } statement is seen as redundant (and harder to define the semanics for) the bare TestExecs (with implied default FlowNodes). Therefore, the group agreed to drop from the FlowNode syntax the ability to describe a TestExec with multiple *execute_stmts*.
- Also on line 178, the form **TestExec NullExec**; (which is allowed only in defining the default FlowNode, outside the scope of any containing Test or Flow) was changed to simply **TestExec**;. This was done to eliminate the need for yet another keyword which would be used in only one context.
- Group agreed to get rid of the "**Title** STRING;" statement in the FlowType block (lines 190 and 384 in the D26, Oct 9, 2008, syntax doc). No one was really sure what purpose this was supposed to serve, and since we already have an id (a Flow instantiated from a FlowType has a name – see lines 199 and 204 of the D26, Oct 9, 2008 syntax document).
- Discuss (again) the need for FuncExec, FunctionDefs, and how ExecResult of Test (TestType) is set when *func_stmt* is used. The need for the FuncExec and FunctionDefs constructs have been the subject of fairly contentious discussions. Not everyone is convinced of the need to include these two constructs, which could result in STIL.4-described test program flows (and tests) being platform-specific and not portable. There are two strong arguments for including them, however.
    - An analysis of the construct hierarchy of many of the languages studied in the development of STIL.4 shows that at least 4 (Verigy SmarTest, LTX Envision, Inovys Stylus, and Advantest OTPL) have a boundary between the Test level (in those languages, an generalized container which itself specifies nothing about the specific type of test to be done) and the FunctionDefs/FuncExec level (a construct which DOES

specify the specific type of test to be executed, as well as the setup and limit conditions used by the specific instance of that test).
    - Many languages (even those that do not draw a distinction between the Test and FunctionDefs/FuncExec levels) support the notion of being able to call user-written procedures either from the FlowNode or from within a test. To support these capabilities, the ability to call an arbitrary function (with arbitrary list of parameters and parameter values, but with a return code specification identical to those specified for Tests and Flows) is necessary.
    - Jim is putting together a series of examples of each of these, and will post it on the web when complete.
    - In the meantime, the current proposal for FunctionDefs and FuncExec is that functions defined in FunctionDefs must return the same values (0 = pass, non-zero = fail) used for ExecResult – and further, that value is used to set ExecResult of containing Test.
- Next Meeting 10/16/08.

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