1450.4 meeting minutes - 10/16/08

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

Not present: Doug Sprague

Agenda:

- Preamble:
  - Record Meeting (*2) (***call not recorded this week***)
    - 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 xxxxxxxx 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.
  - Meeting will be held on Wednesday, October 29, from 10:00 am to 1:00 pm Pacific time, at Advantest's facility (3201 Scott Blvd., Santa Clara, CA 95070). Attendees expected: Bruce Parnas, Ajay Khoche, Jim O'Reilly. Possible attendees: Greg Maston, STARC/SSTAC representatives.
  - Conference dial-in line will be set up and publicized via the P1450.4 email reflector for those who cannot attend in person. Line will be open for the full meeting time; however, a summary will be held during the final hour, to allow people not attending to call in during the final hour only.
- Reviewed changes discussed last time. These changes are reflected in the most current syntax document now available on the website (D26, dated 10/16/2008).
  - Remove "Title" from Flowtype. Instantiated items already have unique IDs.
  - In Flownode statement, we will remove the multiple Exec statement form.
  - Remove keyword "NullExec" from consideration
- More discussion about FunctionDefs and FuncExec. There are two major reasons for considering this additional layer.
  - To allow the separation of the generic test method from the specific mechanism used to implement the test method. In such a case, there could be two or more different mechanisms which would execute a test (for instance, a functional test or a DC test). Allowing separation of the specific choice of mechanism from the generic test framework would allow changing the mechanism while keeping the generic test framework intact. At least two languages being studied (LTX Envision and Verigy SmarTest) draw this distinction.
  - To allow users to call arbitrary user-written functions which behave as tests. At least two languages being studied (Verigy SmarTest and Schlumberger/NPTest/Credence ASAP and its derivatives (XTOS and Credence Diamond language) allow this capability. We feel it's important to permit this in P1450.4.
- The discussion touched on the following:
  - Distinction between FunctionDefs and TestTypes (and comparison across other platforms)
    - Some systems use a preprocessor to convert test language to a C/C++ language.
    - Discussion of OTPL and mapping from FunctionDefs to preheader as an approximate mapping.
    - LTX Envision, Verigy SmarTest have partial descriptions of test behaviors/parameters in FunctionDef-like constructs

  - SmartTest and ASAP allow calling of arbitrary User Defined Functions within a flow node. In dot4 case this would be via the interposed test.
  - Discussion of mechanism for allowing calling of User Defined Functions.
    - Desire to allow test engineer to do this without having to develop a “Test Class”
    - Is it reasonable to wrap all function calls within a TestType wrapper? Jim thinks perhaps not. Ernie is in favor of this approach.
    - Will have to describe the interface between the tester and the function in any case. Return value would have to follow dot4 protocol.
    - Might expect vendor, rather than test engineer, to be integrating these function calls.
    - If we allow calling their functions directly, we’d have to define the conversion between our types and their types.
    - Jim argues that the FunctionDefs allows the flexibility that we want (and that has been lost in previous systems)
    - Bruce says: If we just require everything to look like a Test, it simplifies our work now, and means that we have to require each system to do the wrapping for their User Defined Functions. If we don’t want to care, we can just make the calls from within the text executions and not worry about it.
  - Do we need to define a special-purpose test whose job is to call User-Defined Functions?
    - This exists in Verigy
    - Ajay says that this will still not satisfy the issue of general calling of functions. Doesn’t like the idea of couching these as Tests.
    - In order to define a test that can call an arbitrary function, we’d need to define orderly ways to pass arbitrary parameters, and get arbitrary return values. These return values would need to somehow be mappable to a test result, if desired.
      - We can use pairs of strings {{“12”}, {“Integer”}} to pass parameters.
      - Ernie: That’s fine for literal values, but what about variables: {{“a”},{“double”}}? How will this get converted properly?
  - We’re still trying to define what it means to “call functions” and how to do this in a nondenominational way. We have the several ideas discussed above, but no consensus.
    - Ernie: We have some very basic things (entry points, variables) that are not complete, and time is limited. Is this function discussion where we want to spend our time for Rev 1.0?

- Next Meeting 10/23/08.

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