1450.4 meeting minutes – 06/15/11

Attendees: Paul Reuter, Jim O'Reilly
Not present: Ernie Wahl, Oleg Erlich, Ajay Khoche, Markus Seuring

Agenda:

- IEEE Meeting Preamble (No discussion of proprietary information).
- Discuss semantics of FunctionDefs/FuncExec constructs.

Summary:

- No quorum (only Jim and Paul) – however, we discussed and recommend that:
  - All rules and semantics that we've agreed to for Parameters of TestTypes/FlowTypes (i.e., those regarding the use of and meaning of Const, for example, or whether or not a parameter is passed by value or by reference) should also apply to Parameters of FunctionDefs.

In reviewing this issue, we used as reference the document "FunctionDefs/FuncExec Comparisons" from the "Docs for WG meetings" section of the STIL.4 website, and section 3.9 of the ITC2010 STIL.4 paper (shown below)

3.9 FunctionDefs

In modern object-oriented ATE software, the definition of tests (also sometimes referred to as test methods) can be broadly classified in two ways – those in which the block labeled "Test" specifies what type of test is to be run (for instance, a functional test or a DC test), and those which have a more generic container labeled "Test", from which a specific Test Method (functional or DC test) is executed. P1450.4 supports both models; the FunctionDefs block is the structure used to support the latter. A typical test program fragment from Verigy SmarTest (which uses the "FunctionDefs" model) is shown in Figure 6; the equivalent STIL code is shown in Figure 7.

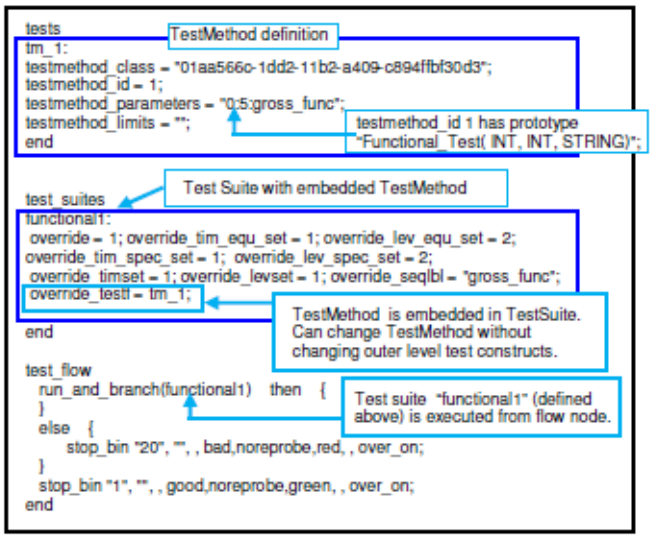

Figure 6. Verigy SmarTest example showing TestSuite and embedded TestMethod

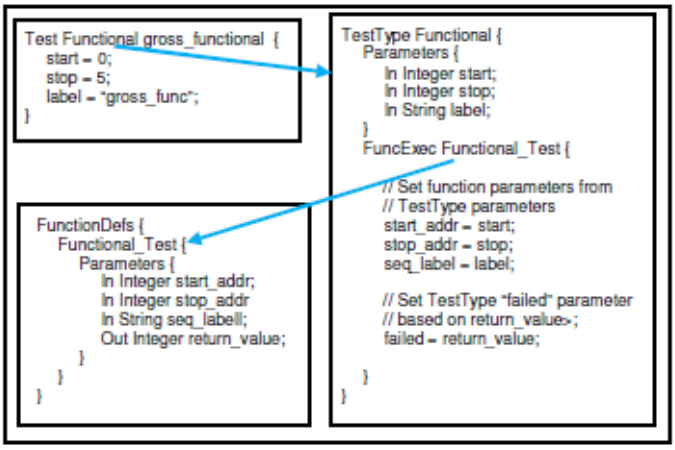

Figure 7. STIL equivalent code (using FunctionDefs) for SmarTest example in Fig. 6.

In contrast, Figures 8 and 9 below show a typical code fragment from Advantest OTPL; in which the TestMethod being executed is intrinsic to the TestType, and doesn't rely on another level of function call (i.e., the FuncExec as defined in FunctionDefs) to execute the test.

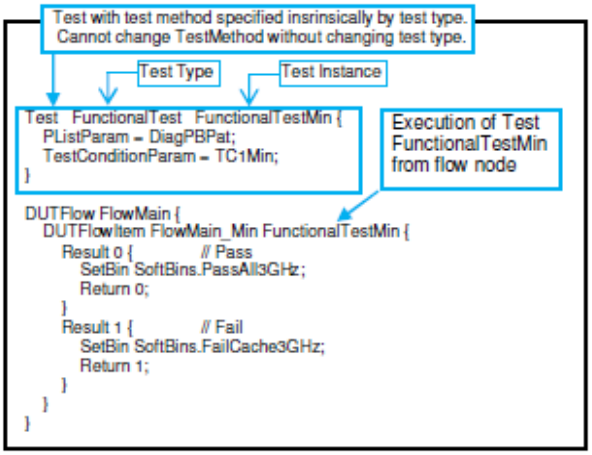

Figure 8. Advantest OTPL example showing TestType with intrinsic Method

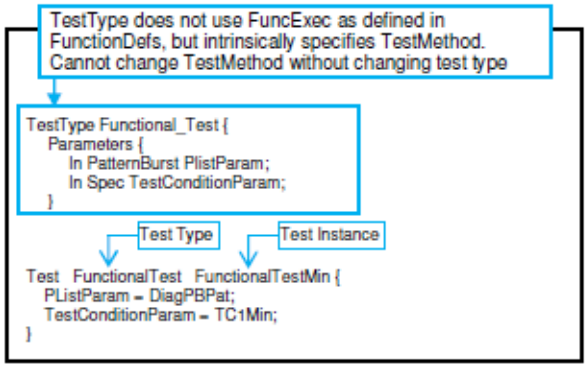

Figure 9. STIL equivalent code for Advantest OTPL example in Fig. 8

Note that in both cases, it's assumed that either the TestType with intrinsic TestMethod or the function defined in FunctionDefs will exist in a library outside the scope of the P1450.4 program, but which can be linked to the program in order to execute the desired TestMethods. Of course, if one is using P1450.4 to translate to an existing tester language, one would simply need to translate either form to the appropriate form for the desired target tester language. We provide both mechanisms for users who wish to target their P1450.4 usage to one or the other of the two models described.

In reviewing the examples, Paul pointed out that the examples were inconsistent – that is, the STIL translation of the Verigy example was incomplete, as it did not include any information about which timing blocks or spec blocks were to be used – which the original Verigy example DOES specify.

While Paul agrees with the idea that it's useful to support an additional level below the level of Test/TestType to specify the actual test code (using a FuncExec statement in the TestType definition, where that function is elsewhere defined in a FunctionDefs block), it seems logical to expect that the TestType parameters for the case in which a FuncExec call is used should include the same parameters as for the case in which a FuncExec call is not used. Additional parameters needed by the FuncExec call must be derived from those passed into the TestType, or the TestType may include optional parameters with default values which are only used if the FuncExec call model is used.

Further, all parameters which specify things like Spec blocks or Timing blocks should be in terms of the STIL blocks, rather than strings, and mechanisms must be in place within STIL.4 to convert a STIL block name into a string – which is already in place, with the .Name operator, which can be applied to most (if not all) STIL block types.

Actions:

- Jim to rework Verigy example from STIL paper so that the STIL translation is consistent with its original Verigy code.

Reference documents (If logged into your google account, can edit. If not, can only view.)

- http://spreadsheets.google.com/ccc?key=0AoKiPr1I9LY9dF95dkhSTVVqOU5GbWJyWFNhY0JPX0E&hl=en
- Namespace resolution examples document: http://docs.google.com/Doc?docid=0AYKiPr1I9LY9ZGY4dmNjNTNfMGZkOGJ2bmZy&hl=en
- Scratchpad spreadsheet: https://spreadsheets0.google.com/ccc?key=tQ93VDnAZ-Cl9RFKpPrPDzw&authkey=COzyro8K&hl=en&authkey=COzyro8K#gid=0
- Scratchpad "Word" doc: https://docs1.google.com/document/d/1zVu2M8nTJsrm0nFbBhiuM8-YRt4ErYqdy_uSa3x3_T4/edit?authkey=CLrgwrsG#

Next meeting: 06/22/11

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