1450.4 meeting minutes – 03/11/10

Attendees: Bruce Parnas, Jim O'Reilly, Ernie Wahl, Oleg Erlich

Not present: Ajay Khoche, Markus Seuring

Agenda:

- IEEE Meeting Preamble (No discussion of proprietary information).
- Minutes from last week are not yet on the web.
- Questions (reviewing agreement from last week):
  - Do we want to define in STIL.4 code the behavior of tests, or only specify the interface to those tests?
    - P1450.4 will specify the interface to tests, and expected behavior (in unambiguous detail!), but will not include detailed code or syntax constructs which would be needed to implement such tests.
  - Do we want to allow for initialization arithmetic involving parameters and /or variables (when a Test or Flow is instantiated)?
    - Yes.
  - When instantiating a Test or Flow, do we want to allow a later parameter to refer to (or use) the value of a previous one to set the value during instantiation, and what value it sees if it can or can't.
    - No. See rules below:
      - When instantiating a TestType or FlowType, values for Parameters (default and instantiation-provided) will be initialized before any local Variables. This includes any Parameters defined in TestBase.
      - When instantiating a TestType or FlowType, Type default values will be assigned before any values provided at instantiation. There is no requirement regarding the order in which Parameters be initialized. Variables will be initialized in the order in which they're specified in the Variables block of the TestType or FlowType definition.
      - When instantiating a TestType or FlowType, values for Parameters (default and instantiation-provided) will be initialized before any local Variables. This includes any Parameters defined in TestBase.
      - Within a Parameters block, there can be no referencing of other items from within that block. A Variables block, however, can refer to items in the Parameters block – and further, a later-defined Variable can be initialized from a previously-defined one.
  - At instantiation of Test or Flow (from TestType or FlowType) what is the order of initialization of parameters? Per C++ rules, initialized in order defined in type (class) definition. Do we want to use that rule?
    - No. See above.
  - Should .Min, .Typ, and .Max fields of a spec variable be modifiable from within STIL.4 code (as we're allowing for .Meas)?
    - No. Not in dot4. It may be revisited at dot4 phase2 or dot5
  - Can you define spec block selector items (i.e., .Min, .Typ, .Max) in terms of other selector fields items?
    - For now, consensus on leaving Min, Typ, Max as constant; .Meas is non-constant and can have a value assigned OUTSIDE spec block definition.

- New issues from this week (from Ernie's email of 3/10/2010)
  1) Global Variables:
     a) We need to be clear about the difference global variables usage in PatternExec and TestProgram. Dot1 quote: "Global variables are set to the InitialValue at the start of a PatternExec and maintain the last-assigned value throughout the execution of the PatternExec."
        i) The question is, how do we reconcile dot1 global variables (specified by an unnamed Variables block (i.e. a Variables block with no domain name) with TestProgram global variables.
        ii) The solution we've leaning towards (but haven't some to complete agreement, as potential side effects remain to be explored) is to have any dot1 global variables (from an unnamed Variables block) continue to behave exactly as they do in dot1 – they'll be reset to their initial values (if specified) at the start of each PatternExec (or any other block that will behave as a PatternExec would – such a functional test, which may take a PatternExec block as a single Parameter, or the components of a PatternExec block as a series of Parameters). Those variables could be readable outside the context of a PatternExec or equivalent block, and could be also writable (perhaps to no effect, since if an initial value is specified, such a variable will be reinitialized at the start of each PatternExec). The exact mechanism for accessing such variables needs to be worked out, and we'll continue to discuss this next week.
        iii) Global variables are defined inside the TestProgram block by using either of the forms shown below

            Variables { *var_elements_stmt** } | **Variables** VAR_DOMAIN)*

            INSIDE the TestProgram block. If the first form, there's no conflict with dot1 global variables (declared OUTSIDE the TestProgram block); if the second, we have an instantiation of a domain-named Variables block, so again there's no conflict with dot1 global variables (again an unnamed Variables block declared OUTSIDE the TestProgram block).

     b) Global variables ought to be able to refer to each other (define before use rules apply) to offer test engineers capability that's available on most testers today (correct me if I'm wrong).
        Agreed – we'll allow this, both for global (TestProgram) variables, as well as local variables defined within TestTypes or FlowTypes. Parameters, on the other hand, CANNOT refer to one another.
     c) I prefer not to use global Variables blocks in TestType|FlowType definitions (see 2a below)
        Agreed – see resolution for 2a below.
  2) TestType|FlowType Variables:
     a) I believe the form "Variables VAR_DOMAIN " on lines 640, 661, and 686 unnecessarily complicates variable semantics, does not reflect how most testers treat variables, and provides little benefit - allow one block only
        Resolved. We'll allow both forms, but allow one block per type definition only. If you want additional variables, simply add them to the single block.

        Changed lines 640, 661, and 686 from
        Variables { *var_elements_stmt** } | **Variables** VAR_DOMAIN)*
        to
        Variables { *var_elements_stmt** } | **Variables** VAR_DOMAIN)

        The Variables statement in the TestProgram block (line 700 of D32) will still allow more than one Variables statement.
     b) I believe allowing more than one Variables block is unnecessary.
        Agreed – see resolution for 2a and 1c above.
     c) I think that TestTypes|FlowTypes may map to functions on many testers where Variables will map to local function variables. I think it's unreasonable to constrain these to "no cross-references".
        Agreed – will allow, but only for Variables (not Parameters). See 1b above.

d) In my opinion, if global and local Variables blocks have different semantics, they should have different type names

Resolved – global in this context was referring to dot1 global variables. Once we get the above issues resolved, this issue will also be considered resolved.

- Open issues - are there other open issues that should be considered? A review of the open issues list can guide us here.
  - Issues list: http://spreadsheets.google.com/ccc?key=0AoKiPr1I9LY9dF95dkhSTVVqOU5GbWJyWFNhY0JPX0E&hl=en
  - Namespace resolution examples document: http://docs.google.com/Doc?docid=0AYKiPr1I9LY9ZGY4dmNjNTNfMGZkOGJ2bmZy&hl=en
  - If logged into your google account, can edit. If not, can only view.
- Next Meeting 03/18/10.

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