P1450.4 meeting minutes - 02/08/06

Attendees: Dave Dowding, Ernie Wahl, Jim O'Reilly, Doug Sprague, Bruce Parnas, Oscar Rodrigues

Not present: SB Thum, Tony Taylor, Daniel Fan, Greg Maston, Jim Mosley, Bob Roberts, Jose Santiago

Agenda/Summary:

- Continue review of Conceptual model (CM) vs. Syntax document (SD)
- Issues discussion:
  - Bypass in PreActions block
    - Intent was to skip around the Body and PostActions and Arbiter to the ExitPort block.
      - Ernie – Would like to bypass not only the above (Body, (common) PostActions, and Arbiter, but also the port-specific post-actions. In other words, bypass all the way to the branch point. As an example of why this is needed, the post-actions (common or port-specific) may take certain actions if the test passed or failed (i.e., incrementing a counter, setting a bin) – and if the test wasn't executed, then I want NOT to do these actions **(Ernie, if I've misstated your thoughts, please correct me! Jim)**
      - JimO – there are two different uses – one is to skip the test, the other is to test or debug branching logic usage. In the latter case, it makes sense to include the Port Actions. (In NPTest/Credence ASAP, you could branch to a specific port with its associated bin block assignment action.) .
      - Ernie wants to skip the entire block including the PortActions.
      - Tony and Greg indicated that the Skip had a integer that could be used so that the if you skip to the arbiter to the ExitActions port with an assigned an integer.
  - TestBase and the yet-to-be-determined contents: Should this capability be provided by the Inheritance statement or using a Defaults mechanism?
    - Ernie feels that both are needed? (Editor's note: need to understand WHY?)
    - Bruce indicated that if Inheritence is the mechanism then you don't need Default.
    - Votes:
      - Bruce votes to use a required inheritance… or a explicite definition in the TestMethod. In favor of Defaults mechanism. Thinks that this Inherit can be clearly defined explicitely
      - Ernie votes for the inheritance behavior. Don't care if the syntax states it. This is a functional requirement not syntactical requirement. Difference whether you place Inherit or not use but always have it
      - Doug votes for inheritance as explicite. If you don't want to use Inheritance then you fill in all the blanks.
      - Bruce wants that if you Inherits then you can get the behavior, or you do not use Inherit you must supply all information required by the TestMethod. The tool that reads this must do the checking to ensure that all information is provided and appropriately given. He doesn't want to say that you MUST use Inherit.
      - JimO: Both Tony/Greg were suggesting this approach that Bruce suggested. (They were thinking of a Defaults approach, but this description would probably be okay as Bruce described.)

- Proposal: Inherit is required and must be explicit. Voted and carries.

- Back to the Rosetta_Stone document review:
  - TestProgram - No discrepancies.
  - EntryPoint/Points – Decided on plural. We still need a UserExtensible mechanism. Syntax subgroup thinks the UserExtensible is doable – will investigate.
  - TestFlow – No disagreement
  - FlowNode – no disagreement
  - TestModule – (decided that we have issues: TestModule/TestMethod)
    - Two issues
      - What to call this construct (minor, but not insignificant)
      - Should this construct include an explicit TestExec statement, or simply execute an unstated (in the syntax) execution method?
        - If the former, I believe that there were some that had issues with a TestModule being able to call something other than the appropriate execution function (i.e., another TestModule or a full TestFlow)
        - If the latter, need a way of defining how an ATE vendor (or user, if users can create their own or extend those provided by the ATE vendor) would actually create such a construct/object, and make it available to the STIL test program
  - TestBase and Defaults
    - Need definition on the minimal required set of behaviors/contents for TestBase.
    - Ernie prefers to have an explicit TestBase (no preference as to whether the inheritance need be explicitly stated or not), rather than using Defaults as synonymous as TestBase.
      - Defaults is a different behavior (TestBase is not alterable by TEs).
      - The intent of the Defaults was to provide mechanism that test engineers could modify the behavior. TestBase for example has an ID item that is left blank and the Library would also not.
      - Oscar feels that Ernie's suggestion is problematic (need to have Oscar elaborate – JO).
      - Ernie: Defaults is a mechanism that allows modification of some certain things from testbase. We don't expect, for instance, to allow Integral TestMethods (which, admittedly, have not yet been defined!) to be alterable by the test engineer. In fact, it's expected that TestMethods would either be provided by the ATE vendor, or by a somewhat centralized "service" group within the tester operations of user companies.
    - JimO: FlowNode base behavior is currently NOT covered by TestBase (that only applies to TestModule and TestFlow) – FlowNode behavior could be covered by Defaults.
    - Bruce: different objects may want to have different base behavior. What mechanism could be used to accomplish this?
  - TestMethodDefs
    - Provides a container for one or more definitions of function definitions/prototypes – essentially a header file.
    - Proposed and voted to change this to TestFunctionDefs.
  - Proposed then and voted to change TestModule to TestMethod. See discussion about TestModule above.

- TestObject vs TestInstances: both are intended to refer to same thing. In CR TestObject
  - Proposed and voted to use in both CM and Syntax to use TestInstances.
- CM shows TaskNode DecisionNode
  - Not needed – can be synthesized from existing elements.
  - To be removed from the CM.
  - Voted and agreed.
- ObjectRef/ModuleRef in CM and TestExec in syntax
  - In FlowNode, use TestExec in both documents.
  - Should the TestModule (now called TestMethod) include a TestExec statement, or rely on an execute method (whose existence is established by TestBase) for that TestMethod?
  - Alternatively, it seems acceptable that another similar keyword (TestFunctionExec?) could be defined and used instead, with the constraint that a TestFunctionExec statement could ONLY call a function defined by TestFunctionDefs.
- PreActions – no discrepancy, no action needed
- PostActions – no discrepancy, no action needed
- SkipPath vs Bypass
  - Use Bypass keyword, but want different syntax options to allow for bypassing only the test, or all the post-actions (both common and result or branch-specific(only resolved to use Bypass.) See earlier discussion on this.
- Arbiter in CM, ExitPorts in SD.
  - In SD, the full set of Boolean expressions in the ExitPorts block of the FlowNode (and, currently, elsewhere) constitute the arbiter.
  - Decision: use ExitPort keyword (in FlowNode) collection of the Boolean expression and a descriptive tag (where to place is to be determined.
  - For the TestMethod and TestFlow will use a different new keyword.
- In Syntax ExitPorts but is a misnomer and so this should be limited to PassActions/FailActions… other opinion was to have an additional action. Agreed that instead of ExitActions and within there PassActions and a FailActions
- ExitAction CM to ExitPorts… Agreed to use ExitPorts for FlowNode
- BinMap and BinNode – no discussion to date.

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