P1450.4 syntax subgroup meeting minutes - 07/25/05

Attendees: Tony Taylor, Greg Maston, Jim O'Reilly, Doug Sprague

Not present: None.

Agenda:

- Discuss background for the benefit of Greg and Doug.
  - History
  - History - summary of past and current efforts at creating a syntax document
  - Review available documents, state of each, and define what we want to accomplish
- Precedence
- Walk through D14 syntax.

Summary:

- Background:
  - STIL .4 is the portion that comes closest to a programming language, rather than a data transfer language (as other parts of STIL are). In part, that's been a source of much of the difficulty; folks tend to think of STIL .4 as an object-oriented language, and seem to (subconconsciously or not) want to relate the constructs that describe the flow to the various means by which such a language could be implemented.

- Assumptions:
  - We will be basing our efforts on Rev I of the conceptual model (at least, until it's superceded by a later revision – which it will be, since there are some deficiencies in the Rev I model that Dave D. and Ernie W. are working on cleaning. Note that the last 3 or 4 versions of the conceptual model, the diagrams are similar or identical; it's the text portions that differ from version to version.

- Goals:
  - We want to develop get bare syntax (EBNF form) and as few words as needed to explain the syntax. Development of more in-depth examples and explanatory material will follow after the syntax is described. The syntax should be a complete EBNF description of the all the blocks needed for test flow, and include tie-ins to other parts of STIL as appropriate. The exmaples and explanatory text should serve to clarify the work – as such, it's somewhat an iterative process, but it HAS to start with the conceptual model, followed by the syntax.
  - Question: Translation vs. interpretation (native execution). Is the language intended to be translated into the target language of any particular tester, or is it intended to be directly interpreted on a tester.
    - We'd like to develop a language that is a modular representation of flow, which can implemented as a directly compiled language on a tester, or translated to the native language of a tester. The language should support both approaches and be "agnostic" in that respect.
    - Stated differently, we want to STIL .4 to be a descriptive language and NOT a means of implementing this on a particular target tester. Nonetheless, we want to structure the language so that it can be implemented on particular target testers using modern programming practices without undue difficulty.

- History:
  - There have been three major, fairly complete, but unreconciled (with each other) versions of the syntax (D11, D13, and D14).
  - All previous attempts seem to get derailed when we run into the issue of whether we were defining a data transfer language or an executable language. The terms and concepts used during discussions seem to lead us down the path of actually considering HOW a system which executes such a language could be constructed.
  - In addition, we have also considered whether concepts from OpenSTAR consortium OTPL language could/should be included. That effort diffused our focus somewhat.
  - Currently, we're working with D14, and will revisit the other two to determine what capabilities (elements), if any, from those drafts need to be included. To go along with those three drafts, there are various use cases and examples that we're using to see how well the described syntaxes (syntaces?) actually perform in use.
  - D11: Created by Tony
    - An attempt to gather the various previous syntax proposals into a common document. Includes the conversion of the Inovys-supplied Stylus example into the proposed syntax.
    - **Tony: Can you recall (and provide) more detail here about what else was new in this draft?**
  - D13: Created by Jim.
    - An attempt to reflect the conceptual model diagrams in syntax. Based on the D11 syntax.
      - Adds explicit test object block, with its unconditional and result-conditional post-action syntax
      - Adds explicit (and clumsy!) "arbiter" syntax to post-execution actions of TestObject and TestNode.
      - Creates common action list for TestObject and TestNode pre-actions.
      - Creates common action list for TestObject and TestNode post-actions.
      - Note thet the pre-action list and post-action list above are NOT common (pre-action list can include (conditional) "Bypass" action – whose behavior for both TestObject and TestNode needs to be clarified.
      - Note also that the pre-actions lists can include Exit, Stop, and Binning actions (either conditional or unconditional). Do we need (or want) these actions as pre-actions? I included them in an attempt to make the set of options among the various blocks which use them as consistent as possible.
    - D14: Tony – can you clarify what was new/different in this draft? I believe it was to create a syntax draft which corresponded to Ernie's cut at casting the Inovys Stylus example into the latest concepts and discussions about syntax.

- Discussion:
  - Tony began a walk through the D14 syntax draft, beginning at the top with the test program block, and walking through that block and down through the various other elements (test flow, test node, test object, test method).
    - It was noted that the Flow block was completely encapsulated within the TestProgram block, while other flow-related blocks (TestMethodDefs, TestObjectDefs) were defined outside the scope of the "TestProgram" block, but were used within the TestProgram block. Is this intentional? Not necessarily – it's one of many areas that we need to revisit, while remaining consistent with other parts of STIL in the area of scoping and namespaces.

- 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