1450.4 meeting minutes – 01/21/10

Attendees: Bruce Parnas, Ernie Wahl, Markus Seuring, Jim O'Reilly, Oleg Erlich, Ajay Khoche

Not present:

Agenda:

- IEEE Meeting Preamble (No discussion of proprietary information).
- Discussion items
    - Discussion of inheritance and parameter overriding. Several proposals have been made (see recent email exchanges); let's select one.
        - Option 1: Using keyword "Override" in place of type name in parameter declaration statement – in both the single form or the block form:

```
Parameters {
   Override failBin = Contact;
   Integer newparam = 1;
}
```

or the block form

```
Parameters {
   Override {
      failBin = Contact;
   }
   Integer {
      newparam = 1;
   }
}
```

        - Option 2: Specifying override values for parameters from base classes in the Inherit statement

```
Inherit TestBase { FailBin = ContactBin; }
```

        - Reviewed structure of inheritance, and the rules for inheritance regarding:
            - Parameters: Derived class included ALL base class parameters plus any defined in the derived class.
            - Action blocks (Pre, Post, Pass, Fail, or ExitPort actions): Each action block (Pre, Post, etc.) defined in the derived class COMPLETELY replaces the same action block in the base class.
        - Discussion back and forth regarding the pros and cons of each approach. One benefit to the Override keyword instead of the Inherit statement form would be that it allows other things (attributes, perhaps) to be altered in the derived type itself.
        - After a discussion of the two options, a vote was taken, and we decided 3-2 in favor of the Inherit statement form.

    - **The next items were not addressed at this meeting**
    - Spec block, Category, Spec variable namespace resolution in Tests/Flows.
        - Allow Spec/Variable/Category hierarchy as well as Spec/Category/Variable hierarchy?
            - Per input from Mentor Graphics and Test Insight (more vendors being polled), Spec/Category/Variable hierarchy should be enough. Disallow Spec/Variable/Category hierarchy.
        - Proposal: Namespace resolution precedence rules.
            - Local variables or Test/Flow Parameters of type SpecVariable

        - Spec variables in any named in context (i.e. Spec block, as well as Category and Selector, if needed, are provided as parameter(s) to Test or Flow).
        - Global variables of type SpecVariable.
    - Need to add data type Spec (for an entire Spec block) to *stil_data_type*. (Done, not yet published on web)
    - Develop, if possible, rules for allowing dot0 compatibility (i.e., what happens if the Test or Flow doesn't specify a spec block name, and all spec/category blocks are developed according to dot0 rules? Do we want to allow resolution based solely on Category/Selector specification - assuming that, as per dot0, all category+variable names are unique. I need to think this one through, but thought I'd put it on the table. As I was thinking through the rules above, it occurred to me that we *might* be able to make this one work also.
  - Discussion of retest proposal (included at the end of the current syntax document).

- Open issues - are there other open issues that should be considered? A review of the open issues list can guide us here.
  - http://spreadsheets.google.com/ccc?key=0AoKiPr1I9LY9dF95dkhSTVVqOU5GbWJyWFNhY0JPX0E&hl=en
  - If logged into your google account, can edit. If not, can only view.

- Next Meeting 01/28/10.

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