1450.4 meeting minutes - 10/02/08

Attendees: Jim O'Reilly, Ernie Wahl, Bruce Parnas, Ajay Khoche

Not present: Doug Sprague, Kevin Coggins

Agenda:

- Preamble:
  - Record Meeting (*2) (call not recorded, since we did not have a quorum)
    - To listen to the meeting recording, do the following:
      - Call the (US) dial-in numbers 1-877-421-0003 (toll free) or 1-770-615-1374 (toll)
      - Enter the passcode code 747464
      - Once dialed in with the proper access code, enter *3 (star 3)
      - Then enter the file number 30870001 for this conference (this number will change each week).
  - Press 1 to listen to the conference
  - IEEE Meeting Preamble (No discussion of proprietary information).
- Poll WG members to see who will be at ITC. Would like to arrange a face-to-face meeting if possible.
  - 3 of the 5 WG members will be at the conference or in Santa Clara during the conference (Bruce, Ajay, and Jim). Greg Maston from Synopsys will also likely be at the conference.
  - Bruce agreed to host meeting at Advantest. Since representatives from STARC/SSTAG in Japan will also be attending the conference, and want to meet with us regarding progress on P1450.4 (and other STIL activities), I'll contact them and see if they'd like to participate.
  - Jim, Bruce will work out the logistics, and publish the meeting time/place, and agenda.
- Review changes to Variables/Parameters blocks and finalize syntax and semantics. Updated syntax document and StdTypeExample will be posted on the website prior to the meeting.
  - Specification of optional parameters for TestTypes or FlowTypes. To specify parameters that are optional, the keyword None (based on Doug's suggestion – Python uses this keyword for an identical purpose) is permitted, in addition to other valid values. Therefore, the following is permitted:

```
TestType StdFunc {
    Parameters {
        In Category timcat = None;
        In Selector timsel = None;
        In Timing tim;          // Required, therefore not initialized
        In Integer delay = 0;
    }
}
```

    If a parameter is initialized to None, a value is not required (but can be provided) when the type is instantiated. If a value IS provided when the type is instantiated, then the value provided at instantiation is used; otherwise the value None is used. When executed, any parameter that has the value None is ignored by the underlying run-time code. Ernie would prefer = { Opt; } (as shown below) instead of = None; but is OK with None, as long as the semantics are clearly defined.

```
TestType StdFunc {
    Parameters {
        In Category timcat = { Opt; }
        In Selector timsel = { Opt; }
        In Timing tim;          // Required, therefore not initialized
        In Integer delay = 0;
    }
}
```

    This is in contrast to parameters that are NOT initialized (either to None, or to a valid value). Such parameters MUST have a value provided by the user at time of instantiation of the type (TestType or FlowType).

- Array initialization.
  - Discussed syntax for array initialization. We considered 3 cases:
    1. Initialize all elements to the same value.
    2. Initialize each element to a distinct value.
    3. Partial initialization (allow initial value list to be smaller than or the same size as the array. If initial value list length < than size, then last element is used for remainder of elements). This is a more general case of the "intialize all elements to the same value" case.
  - We discussed each of these, and decided to support cases 1 and 2, but NOT to support case #3.
  - What if uninitialized variables are accessed? What's seen? Is it legal? Need to check with Greg Maston on this.
  - Syntax changes needed to support cases 1 and 2.
    - Case 1: Add line 235 combining LHS of 234 and RHS of 233. Initialize all array elements to the value 12.

      Integer myInitializedIntArray[4] = 12;

    - Case 2: On line 234, use [ ] around initializer list, and make length optional. The following two forms are equivalent. Since the initializer list also carries array length information, the length specifier on the left-hand side is optional (but HIGHLY recommended).

      Integer myInitializedIntArray[4] = [12,0,0,0];
      Integer myInitializedIntArray[] = [12,0,0,0];

- Array usage in action statements (line 65). Discussed whether to allow assignment of one array to another in one statement, rather than assigning element by element. For instance, for an array of length 5:

  myIntegerArray[ ] = [1];

  or

  myIntegerArray[ ] = [1,1,2,3,4]

  Discussed this, and decided against supporting this behavior. Instead, if, in an action statement, you want to assign values to multiple array elements, or copy one array to another, you must do it element by element.

- Conversions between integer and boolean types, between integer and TestStatus, between boolean and TestStatus. Based on Summary:
  - Integer to boolean: 0 = True, anything but 0 = False
  - Boolean to integer: True = 0, False = 1
  - Integer to TestStatus: 0 = Pass, anything but 0 = Fail;
  - TestStatus to Integer: Pass = 0, Fail = 1;
  - TestStatus to boolean: Pass = True, Fail = False
  - Boolean to TestStatus: True = Pass, False = Fail

- Next Meeting 10/09/08.

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