P1450.4 meeting minutes - 11/12/03

Attendees: Dave Dowding, Eric Nguyen, Ernie Wahl, Don Organ, Jim O'Reilly
Not present: Jim Mosely, Jose Santiago, Yuhai Ma

Agenda:

- Review of minutes from previous meeting.
- Review email from Dave - (yet-again) updated flow diagrams.
- Proposal about how we our flow extension constructs link into the remainder of 1450. P1450.1 environment block used by P1450.6 WG to place their extension information.

SUMMARY:

Flow-node module contents:

As we did last week, we continued discussion about the types of objects that the body (module) of a flow-node could contain. Dave has updated (yet again!) the flow diagrams (see link below);

> Updated the Flow Diagrams (specifically Figure 2A) to reflect the question raised in Jim's meeting notes from last week's conference call.

we used these diagrams as a basis for discussion. Also see the figure on page 2 below.

In the discussion of Fig. 2 (Fig 2a - choice 1 and choice 2), Ernie pointed out that choice 2 (since it is a test block, not a flow-node) should just have pass and fail exit actions, which (as shown) converge to a single exit point. Dave will update the diagram to reflect this. Ernie reiterated that, in his view, the body of flow-node is not an instantiation of a test, but should just be a **pointer** to a test method or a test.

Dave put forth an alternate proposal: To reduce the levels of complexity, what if, instead of the extra layer of a test, we simply pointed, in a flow-node module, to another flow-node (i.e., a subflow with just one flow-node). Ernie felt that, while such an arrangement might be workable, it could also burden the syntax for the flow-node construct with additional elements (to allow it to serve double duty, as it were) - and also felt that it would blur the distinctions between a test block and a flow-node.

In considering the distinction between a flow-node and a test (each of which has pre-actions, post-actions, an arbiter, and exit actions/paths), Don felt that, for the types of SW systems he's worked (and is working) on, the additional segmentation of function, while it didn't necessarily add any additional capability, it also didn't get in the way of things that he would like to do. So, if the defaults for actions/arbiter/exit paths for either flow-node or test are chosen intelligently, he's amenable to the concept of having both a flow-node object and a test object (we **really** need a more descriptive name for this object).

I had the sense that, at the conclusion of the call, all present felt that, if we could develop reasonably clean syntax for the concepts represented in Fig. 2a, choice 2, and Fig. 2b, they'd be agreeable to that representation (I hope I haven't misrepresented anyone's postion!).

I also came away from the call with the feeling that it might be time to revisit some of the Don's and Ernie's past efforts at developing a syntax, and update them based on our current understanding of the objects (flow-nodes, test blocks, test methods, subflows).

To aid in understanding how currently available testers deal with these constructs, Jim O. will put together a summary from as many different testers as possible showing how their flow and test block constructs map onto the constructs we seem to be agreeing on.

Links to other STIL constructs:

We also need start defining how we link our flow extensions with the rest of STIL. In draft 15(?) of P1450.1, clause 17 describes an environment block. Could we define an environment block called "test program flow?". For an example of how the P1450.1 environment block is used, see P1450.6. In the drafts of that standard, CTL data is placed in a P1450.1 environment block.

The environment block may not be the best place to reference flow information, but it could be used for this. Dave will forward to Don copies of the current .1 and .6 drafts, to be placed on the web site for reference.

In any case, we need to start thinking about where the test flow (and above that, the test program) would get referenced in STIL, and where other STIL constructs (such as Category, Selector, Timing, DCLevels, DCSets, PatternBurst - i.e., the contents of the PatternExec block) will be referenced.

That's about all for now - let's pick this up again on Wednesday.

Jim