P1450.4 meeting minutes - 10/15/03

Attendees: Jim O'Reilly, Doug Sprague, Ernie Wahl, Dave Dowding, Yuhai Ma, Jose Santiago, Jim Mosely

Not present: Don Organ

Agenda: Review Dave's latest diagrams (sent out 10/10/03). Get closure on concepts from flow diagrams.

SUMMARY:

Main issues of discussion centered around the model of the flow-node, and the relationship between flow-node, the "module" part of the flow node (see Fig. 1 in the PDF file of diagrams that Dave sent by email on 10/10/03) and the subflow or subflow node (see Fig. 2 in the same document).

Another issue was the interchangeability of "subflow" with "test" in the module of a flow-node. If a subflow can substitute for a test in a flow-node module, then it must have only one exit path - which is in direct conflict with the idea that a flow-node can have multiple exit paths.

The big question seems to be how many different types of objects do we have, want, or need? Here's what we've got so far:

Flow-node: Everyone seems to be comfortable and in agreement about the conceptual model of a flow-node, consisting of an entry point, pre-actions(s), a flow-node body, post-action(s), an arbiter, and one or more exit paths, each of which has its own set of exit actions.

Subflow:

There's no agreement yet on what a subflow consists of. Is it a separate type of node, similar to a flow-node, but whose body (the "module", from Fig. 1) is a set of flow-nodes instead of a simple test method? Or is it simply a collection of flow-nodes? If the former, then the diagram shown in Fig. 2 is fairly accurate, and a subflow is a collection of flow-nodes wrapped by pre- and post-actions, an arbiter, and exit port actions.

One of the main points of discussion is about how a subflow could be used in place of a test in the module of a flow-node (something that Ernie needs). In that case, a subflow is simply a test whose method is "subflow". If a subflow is to substitute for a test (which has only one exit path - see discussion below) then it must also have only one exit path - so that requirement conflicts with the requirement that a flow-node can have multiple exit paths.

I suggest two possible alternatives:

- A subflow is simply the same as a flow (a collection of flow-nodes), and we drop the notion that a subflow has pre- and post-actions, an arbiter, and sub-flow exit actions. The exit paths from the subflow would simply be the exit paths available from any of the flow-nodes in the subflow (which can be connected however the user needs them to be - so there can be any number of exit paths from a subflow.

  This alternative has the advantage that a subflow and a top-level flow are really no different conceptually - the only difference is that top-level flow(s) have predefined name(s) (or type(s), or some other attribute which identifies the flow as a top-level flow instead of a subflow). The disadvantage is that with many exit paths, such a subflow cannot directly substitute for a test in a flow-node module - because the flow-node module can have only one exit path. (This could be worked around by adding a final flow-node with only one exit path to the end of the subflow - and all exit paths of the remainder of the subflow feed into the input path of that final subflow flow-node).

- We define a subflow as a particular entity which consists of one or more flow-nodes, preceded by a pre-actions block and followed by a post-actions block, an arbiter, and one or more exit actions (exactly as shown in Fig. 2 or ???). Since a subflow can have one or more exit paths, I think that such

a subflow could substitute for a test in the module of a flownode ONLY if that subflow had one and only one exit path (or we provide some syntax to direct an exit action to flow to another exit action in the same subflow).

During a very longish conversation with Jim Mosely about this issue on 10/16/03, we came to the conclusion that many of the diagrams shown in Dave's document represent valid use models - in particular, figs. 4 and fig. 5 seem to be the most accurate, and whatever syntax we come up with must support both use models.

Test:

We haven't talked much about this entity recently, but I think that most of the group feels comfortable with the notion that a test is simply a call to a test method. However, at ITC 2002, we had come up with some diagrams for a test which included pre- and post-actions, an arbiter, and exit paths, including exit path actions. In that diagram, the main difference between a test and a flow-node was that a flow-node could have multiple exit paths, and a test had only one exit path, and two exit actions - a pass action and a fail action, which both continued through the single exit path. See the diagrams in http://65.119.15.228/stil.4/flow_diagrams_100702.pdf for the details.

Flow:

A collection of flow-nodes. There has not yet been much discussion recently about the details of this - except that I think most of us feel comfortable with the idea that there can be multiple top-level flows (multiple entry points, such as a main flow, an initialize flow, a reset flow, an error-handling flow, etc.)

Test program:

The test program would be a collection of all the pre-defined top-level flows (entry points) that were discussed above. A workable description of the test program block can be found at the bottom of p5 of http://65.119.15.228/stil.4/1450_4.pdf (the original P1450.4 document from 1999 that we frequently allude to in our calls).

Stream-of-consciousness transcript:

I tried to capture as many of the discussions as possible, based on my notes taken during the meeting and also relistening to the recording of the meeting. Not necessarily in completely chronological order - nor guaranteed to be completely accurate.

Review Dave's latest diagrams (sent out 10/10/03).

Jose Santiago: Any progress on STC OpenSTAR presentation from Yuhai? (Yuhai is planning to share information regarding STC OpenSTAR flow definition). Dave: no further progress.

Dave: Jose - tell Yuhai what you were thinking about when we talked about the presentation. Dave had stated that Yuhai and Eric (from Advantest) - being able to leverage between STIL and STC OpenSTAR

Jose: discussion was to have Yuhai share what STC is doing for OpenSTAR flow definition (with an eye towards leveraging efforts.

Dave: go through today's discussion about the models we've been talking about, that may be a natural place for Yuhai and Eric update us on STC OpenSTAR flow definition.

Yuhai: Let's go through diagrams of STIL work so that I can get up to speed.

Dave: Changed diagrams - Added informative information (previously discussed but never captured) to diagrams (pre-, body, and post). First object or model that we tried to get some terms around was the notion of a flow node, have a pre-, a body, and a post- section to it. In diagrams, "common post-actions" changed to "post-actions". "post-actions" for ports 1-N changed to "exit actions". Input arrow, output (exit path) arrows, and skip path (if directed by pre-actions, can flow directly to post-actions block). Can direct flow around body or module and go directly to a specific exit-action block.

Ernie: OK with labels - might be useful to understand what goes on behind the labels, maybe at some later point, we can look at the labels again and decide if labels reflect what actually goes on wants to look at what's behind the labels, names may change later if actions warrant.

Jose: OK with changes - makes diagrams more clear - like pre- and post-actions, in both main flow-node diagram and subflow diagram.

Dave: Had been using "port" instead of exit action, but at Jose's suggestion, dropped the use of "port", since it is an overused term - use a more specific term (i.e., exit actions).

Ernie: Relationship of flow node to subflow - Fig. 2 - coming out of subflow, will exit actions join? Module in Fig. 1 can be a simple test or a subflow. Module does not necessarily contain a test, but simply a pointer to a test (or subflow). Want to be able to edit one test definition, and have it be changed everywhere it gets used. Wants interchangeability between subflow and test in module body. Pre-actions and post actions are generally done based on test type, not based on where we are in the flow.

Slight difference in exit actions for flow-node and subflow-node. Flow node has multiple exit paths, subflow has one exit path. Body of a flownode can be a subflow, but body of a flownode (if a subflow) is not a flownode in the samea flowcan be a flow-node

What's different between a subflow and a flow-node? Are there differences? Module does not directly contain (an instance of) a test or subflow, but simply points to a test or subflow. Diagram labelled flow-node is called

Subflow is not a node in the graph, it's merely being pointed to by the module of a subflow.

Dave: If there is a construct difference, between flownode and subflow, need to show it.

Ernie: Is Fig. 2 simply Fig. 1 with the module part blown up? If so, then Fig. 2 should remain labelled subflow node. However, as Ernie sees it, a subflow is not a flow-node, but rather is a test whose method is "subflow".

Dave: Do we need subflow pre-action and post-action, as well as flow-node pre- and post-action?

Ernie: to keep whole picture coherent, the entity that module 4 points to (assuming it points to something, rather than containing an instantiation) can be either a subflow or a test. Want to have post-actions and pre-actions NOT based on where you are in the flow, but based on what kind of test was run (i.e., VOH test is always binned the same way, instead of having different VOH tests binned differently based on where we are in the flow).

Dave: Longish discussion about about various aspects of relationship between subflows and flow-nodes.

Jim O: Stop action is a particular type of flow-node, NOT a characteristic of a general flow-node. Adamant about bins not being terminal flow nodes. Bins are (optionally-assigned) attributes of flow-nodes (assigned by input- or exit-actions); Terminal flow nodes are not bin nodes. I see (at least) two special types of flow-nodes - a terminal flow node that has an input but no output, and a begin flow-node that has an output but no input.

Jose: Fig. 1 is model - figs 2-4 are use cases. Add stop definition to Fig. 1? Should stop-actions be part of the flow-node as well as the subflow?

Dave: question on the table as to whether we really DO have a subflow node (at one point in our previous discussions, we had different constructs for subflow that we did for individual flow-nodes).

Jose: 3 levels (top-flow, a form of subflow, then a subflow which contains a number of flow-nodes; exit or stop may not fit well in flow-node, but may fit with subflow)

Dave: different constructs for subflows than for flows? Where does test flow (and test program) information fit in the overall STIL data hierarchy? CTL has allowed for the ability to define different environments (test modes?) - CTL blocks are contained within the .1 environment block. Need various subflows (init, reset, etc., including characterization).

Structural differences between subflow and flownode?

Yuhai: Is there a high-level main flow (the top-level container).

Ernie: If we have flow-node and subflow, if defined in a consistent manner, then test flow is simply the top-level subflow (i.e., in C, main( ) is just a regular function).

Jim O.: Original documents from 1999 had test program block structure, with various types of subflow (main, init, reset, on error, etc.). Concept of test-program block is useful - we should keep it. Tentative agreement on this point from others.

Ernie: Subflow is special kind of test - one entry and one exit. So, subflow will NOT have multiple exit paths.

Doug: Felt that it would be desireable if a subflow and a flow were similar conceptually.

Jim Mosely: top-level flows and subflows have same behavior - want maximum flexibility - don't show stop bubbles attached to exit arrows from flow nodes of subflows.