P1450.4 meeting minutes - 01/12/05

Attendees: Dave Dowding, Tony Taylor, Tom Micek, Chris Nelson, Jim O'Reilly, Don Organ, Ernie Wahl, Doug Sprague, Yuhai Ma

Not present: Oscar Rodriguez, Jim Mosley, Eric Nguyen, Dan Fan, Jose Santiago

Review of Chris Nelson's multi-site requirements doc:

- The bulk of the meeting was devoted to discussion and explanation by Chris Nelson of the Intel multi-site requirements document provided by Chris. The following points were noted:
  - Intel uP does 2X multi-site (both DUTs are the same) for structural test on CPU products – can theoretically scale to > than 2.
  - Users program for single site – bulk of parallel test activities are handled by Test class (test method) programmers.
  - Definition of terms: "Multi-DUT" – means same types of devices, not different devices. Different types of devices would probably require different test programs – if you want to test them on the same computer, a multi-threading approach, with each thread running a different program, is probably appropriate (each test program is completely independent, running in its own context). For example, a product which has several different die types in the same reticle.
  - Testing multiple devices on one control computer. Test program is loaded on a single computer, and is testing multiple devices. OpenStar architecture DOES allow for multiple control computers – in that case, each test program is loaded on a separate control computer, and degenerates back to the single DUT case.
  - Test programming language is OTPL (OpenStar test programming language). Analagous to STIL constructs – a source declaration language (flow components, timing, levels, etc.) rather than a programming language. At that level of programming, there should be very little, if any, multi-DUT programming dependencies. Exceptions: markers in flow to deal with branch priority, and obviously, multiple sets of DUT pins (which end up in socket file).
  - Several permutations:
    - Single control computer, multiple DUTs, independent test instruments for each device (no sharing of instruments across DUTs). Some modification of the tester OS is needed t
    - Multiple control computers, multiple DUTs, independent test instruments for each DUT (no sharing of instruments across DUTs). This is much like the case of a single control computer and a single DUT (no tester OS modifications are needed).
  - Ernie: Are mutiple control computers physical computers, rather than logical (VMWare?) computers?
  - Chris: Yes – in today's OpenStar implementation, each computers is an independent Windows computer. There ARE some circumstances – such as offline mode – where the concept of virtual sessions is used – which again degenerates to the single DUT case. If you allow the idea of logical computers (to the extent that the OS running on the physical computer can handle it!), then the situation is much like the case of multiple physical computers and multiple DUTs, and you don't need anything in this presentation. This presentation (design requirements) is based on the assumption (and Intel's experience!) that virtual computers will not scale up to the level needed – and therefore some support in the tester OS (and programming language) is necessary.
  - Ernie: What are modules labelled M1-M4?
  - Chris: Could be anything – digital channel cards, power supplies, etc. Each module can be controlled by only one computer (OpenStar requirement). There's no provision for, say, half the resources of a test module being controlled by one control computer, and the other half being controlled by another control computer. However, it IS allowable for some of the resources of a particular module to be split among multiple DUTs, as long as the module itself is connected to only one control computer. In this case, it's the

responsibility of the test class (test method) programmer to make sense of this configuration. See the diagram on P4, example A, of Chris's slides for an example of this.

- P5 of slides – multiple control computers, but shared instrument (one instrument shared among multiple control computers). Reason for not replicating such an instrument: economic (instrument may be expensive), logistical (can only fit one of these types of instruments into the tester). Intel's message to Advantest (and STC). Don't do anything to prevent this, but don't solve the problem now.
- P8 of slides – terminology.
  - Tom Micek – what does "On Hold" mean?
  - Chris:
    - answer is product-dependent – see P12 of slides. The same thing applies to the "Disabled" state (disabled means that site will not be tested again for the duration of this flow execution – i.e., perhaps device failed and was binned, doesn't need to be tested until a new set of devices is inserted into sockets).
    - Bottom line – the entry into and exit out of "On-Hold" state, and entry into "Disabled" state (and exit out of "Disabled" state – is this possible or meaningful?) needs to be customizable on a per-device-type basis.
    - OS needs to manage the active/on-hold states as it progresses through the flow, and pass the right set of enabled DUTs into each of the test instances. User needs control of this management of active/on-hold states. At each potential branch point, user needs to specify (in the flow syntax) branch priority for which results should be handled first. User also needs to be able to specify flow-convergence points.
- Many of the test program variables and data structures will need to be replicated per-site (search results, datalog streams, etc.)
- Doug – to summarize p10-11, there will need to be facility built into ATE OS to
  - allow both parallel and serial testing and providing the user mechanisms to make that happen,
  - to get back information to know what has happened, and use that information in test program decision process.
- Chris – Yes - burden is on ATE OS to provide these facilities, with the test classes (test methods) making use of these capabilities. Very little multi-site-specific info appears in the user-level test program.
- Many of the concepts on P11 map closely to what STIL .4 has already considered, but there are a few items listed here (i.e., per-pin fail memory), we haven't considered – but probably should.

- To summarize:
  - There appears to be good alignment between what Chris's document outlines and what we've already considered for STIL 1450.4.
  - However, Chris's document does go beyond what's been considered for .4 – it isn't too concerned with boundaries between the flow and the test class (test method).
  - Much of the multi-site smarts would be in the test class (test method) – which is probably a 1450.5 issue.
  - The test flow (1450.4) will need to include some multi-site specific concepts (mostly those concerned with branching and reconvergence in the flow).