1450.4 meeting minutes – 10/01/09

Attendees: Jim O'Reilly, Bruce Parnas, Ajay Khoche, Ernie Wahl, Markus Seuring
Not present:

Agenda:

- IEEE Meeting Preamble (No discussion of proprietary information)
- Minutes from last week's meeting (9//24/09) are now available on the web. Updated syntax document (version D0.28, dated 9/24/09) including the binning changes (see last week's minutes for details) is also available on the web.
- Discussion of spec/category/variable scoping proposal (see "Spec/Category/SpecVariable Handling Proposal" on web under "Docs for WG Meetings"). Can we settle on allowing both opt 1 and opt 2 (with the user choosing which model to follow?).
    - Spec block name required or optional? Ask Greg.
        - Greg's answer – the intent was for the spec block name to be optional. One unnamed, and multiple named spec blocks are allowed.
    - Long discussion about opt1 vs. opt 2. No clear consensus yet, but the pros and cons of each were discussed, and are summarized below.
        - Opt 1 (strict dot0 model):
            - opt 1 allows direct compatibility with dot0 (and more to the point, with some testers whose software follows that model), at the expense of greater encapsulation of data that opt 2 provides.
            - Opt 1 would allow direct translation of spec blocks and the category names within those blocks to those types of testers - with no modification of category names, a key point.
            - However, opt. 1 falls short in the key area of encapsulating data within the spec block - a concept that was considered in the original discussions for dot0, but not implemented. In dot0, the spec block is a syntactical container only, and carries no weight with respect to scoping of categories and variables. In effect, the spec variables in dot0 are global in scope, and the combination of category name + spec variable name MUST be unique. This may make it difficult to combine spec blocks from multiple runs of tools, in which spec blocks may use the same names for categories and variables, but the values of those variables may be different.
            - In addition, in the dot0 model, the variables in a given category can be defined in multiple spec blocks, rather than requiring that the variables within a single category all be defined within a single spec block.
        - Opt. 2 (enhanced model for dot4)
            - Opt. 2 allows for encapsulating the category/variable definitions within the spec block. In this model, the combination of spec block + category name + spec variable name must be unique (in contrast to only the category name + spec variable name of the dot0 model), and all spec variables contained within a single category MUST be defined within a single spec block.
            - Some issues in translating such a model to tester software that follows the dot0 model may occur. In these cases, there are two solutions:
                - User level – insure that the multiple spec blocks that occur within a program do not reuse category names. In this case, the spec, category, and variable names of the STIL program can be translated directly (with no renaming) to the target tester software.
                - If the above constraint is not met, a simple name remapping by the translation tool can be done. The key is to make the category name unique – which can be done in any number of

        ways that still retain the intent of the original STIL program. Two such mechanisms are:
            - Create a new category name based on the combination of spec block name + category name.
            - Append a alpha or numeric tag to the category name (such as “category_name_1”) to the first occurrence of a specific category name. Subsequent occurrences would then use “category_name_2, category_name_3, etc.)
    - There is already precedent in the dot4 standard for such renaming when translating to target testers – and that can occur if multiple bin axes are used in the SoftBinDefs and HardBinDefs. In those cases, when translating to almost all existing systems, the hierarchy of the soft and hard bin defs would need to be flattened, with new bin names based on the combination of axis name and bin name. This situation is not completely identical, since the bin constructs are newly introduced in dot4, whereas in the case of possible renaming of category names, that operation would introduce a change to an existing part of the STIL standard.
    - Finally, since dot0 already imposes the constraint of category name uniqueness, there is no possiblity that accepting the dot4 model would break existing code (which, by definition, MUST adhere to the dot0 model). Likewise, if translating existing code from testers that adhere to the dot0 model back to STIL, there is again no possibility of breaking that existing code.
- In short, allowing opt. 1 along with opt. 2 (with the user specifying which model to use by providing an additional option to the STIL Flow 2009; statement) would provide a minor benefit, but would introduce substantial complexity in both parsers, and in a reader’s ability to understand which set of rules would govern the behavior.
    - This would be particularly apparent if a tester were to implement dot4 as a native software (rather than simply translating STIL dot4 code to the native language of that tester). In such a case, the run-time software of that tester would be revised anyway, and could take the dot4 rules into account.
    - If a tester were to use dot4 as a runtime language, and if both the dot0 and dot4 rules could be allowed (with the program code specifying which was in effect), then the issue of what happens if a piece of code using the dot4 rules (which allow for repeated category names in different spec blocks, and which includes the spec blocks as part of the scoping hierarchy) were run with the rules then changed to dot0 rules (again, using an option to the STIL Flow 2009; statement) would have to be addressed – and it’s felt by members of the WG (who all have experience in both writing and using such software) that such a situation, could be solved, but would introduce substantial complexity.
- Therefore, to summarize, the decision is to allow the dot4 model only. This provides a cleaner long-term path, and adds minimal complexity to STIL translators which would have to take code with the dot4 spec/category/variable hierarchy and translate it (using simple name remapping schemes) to target testers which adhere to the dot0 model.
- The changes to the spec/category/variable hierarchy from dot0 to dot4 can be summarized as follows:
    - No distribution of spec variables among multiple spec blocks. All variables belonging to a single category MUST be contained within a single spec block. NOTE: This is a recommended, though not enforced, practice even in dot0.

        - A category name can be reused in multiple spec blocks – in contrast to dot0, which specifies that a category name MUST be unique. That constraint arises because in dot0, the spec block has no semantic weight – the spec variables (unique in category_name+spec_name), are global in scope. Dot4, on the other hand, does give semantic weight to the spec block name. All variables must be unique in the spec_block_name+category_name+variable_name – and such a variable is global in scope, as would be the truncated form (minus the spec block name) in dot0.
        - Just as spec variables in dot0 can be referred to by the form

          ```
          category_name.spec_name.selector_value
          ```

          (i.e., `my_category.my_var.[Min | Typ | Max | Meas ]`),

          so can spec variables be specified in dot 4 by the form

          ```
          spec_block_name.category_name.spec_name.selector_value
          ```

- Discussion regarding whether or not to use the "equal sign" (=) in the syntax in particular areas.
    - Presently, in the syntax document, properties (of bin axes or of soft or hard bins, for instance) are set without requiring the use of the "=", whereas variable or parameter assignments do use the "=" sign. Ernie's parser currently uses the "=" sign for both.
    - Other parts of STIL provide no strong bias one way or the other. In some case, the "=" sign is used (creating signal groups, for instance), and in other cases, the "=" sign is not used (for instance, in setting properties of a signal, such as base, alignment, or the default sequence lengths for a ScanIn or ScanOut signal).
    - After discussion, we agreed that the syntax document model would be followed
        - Setting of properties will NOT use the "=" sign
        - Variable and parameter assignments will use the "=" sign. Ernie will update his parser accordingly.
- Discussion of retest proposal
    - Did not get to this. Will discuss it next week, and/or via email.
- 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=pEI1-gPUmt2ZTw_kcCTgnKw&inv=jim_oreilly@ieee.org&t=9330484534885518718&guest).
    - If logged into your google account, can edit. If not, can only view
- Other?
- Next Meeting 10/08/09.

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