October 23, 1986

Professor Tony Hoare
Department of Computer Sciences
University of Texas at Austin
Austin, Texas

Dear Tony:

Here are my comments on "Formal Methods in Software Engineering." Since I am unsure of the intended audience, I assume that you are pitching it at the right technical level. It is written extremely well; there is no *need* to change it greatly except perhaps by adding a summary section. The following comments *may* be taken into account if you are intent on preparing a revision.

I believe that the text is structured as a sequence of subsections: Goal of an Engineering Project, Capture of Requirements, The Design Process, Some Successful Research Efforts, Summary (unwritten). I suggest adding some such heading to each subsection to better display the structure.

Goal of an Engineering Project (pages 1,2): Beautifully written. P.2, first paragraph, is a little confusing. Are you talking about arbitrary engineering projects or software engineering? I assume it is the former. Then "guarantee correctness" and "proof techniques" should be replaced by words appropriate for the more general case, such as "sufficient confidence in the design to perform reliably under the given specification." Also, experimentation is a major part of any engineering design and should be mentioned here. (It is not clear if you do or do not advocate experimentation in software engineering design. On p.4, last paragraph there is "...without recourse to...experimentation." In p.4, point (4), there is "...early prototyping and other experimentation on design." My position is that experimentation is to be allowed, and even encouraged, for those aspects-- performance, for instance--for which formal manipulations are, as yet, inadequate.)

Capture of Requirements (P.3 and first two paragraphs of p.4): How about settling for a tired word, "specification"? P.3, first paragraph, is brilliant. May I suggest that *and* is the proper connective for *enumeration* of desired properties? (It is also the connective of choice for decomposition, both for design and understanding of the specification.)

P.3, second paragraph: You may elaborate on the need for designing an
appropriate algebra for each application. On the one hand, the formalism should allow for nontrivial formal manipulation and on the other, the elements of the algebra should have reasonable interpretation in the application domain, so that the validity of the specification can be checked. Design of appropriate algebra for an application is itself an engineering project.

Is it appropriate to talk about the desirable properties of algebra/specification? How about decomposability? P.4, second paragraph: One goal of specification is to allow initial exploration into the mathematical properties of the problem.

You don't include performance as a part of specification. It may be useful to state it and also, report that formal manipulations without experimentation have not been successful in satisfying this aspect of the specification.

p.4, first paragraph: explanation of Scott's work, in this context, is slightly confusing.

The Design Process (last paragraph of p.4 and p5): you leave open the possibility of having a multitude of notations, one appropriate for each stage. Do you really mean it? (Is CSP a notation which permits refinement from specification to implementation? Are there other notations? I know of only one more!). It may be worthwhile to point out that the design should be "minimal"--the least defined which meets the specification. This is to allow for evolution over time.

P.5, point 1: replace "calculation, proof techniques" by "formal manipulations of formulas."

P.5, point 2: "efficient description" is not what you have in mind. "Runnable" should be "executable."

Some Successful Research Efforts (p.6,7): This is always a difficult part to write: it is easy to gladden the enemies and sow confusion among friends.

I suggest including some mention of CCS and temporal logic. (Do you know of O'Donnell's work in equational programming?) I am unsure about Gypsy's contributions, for inclusion here. Some mention of polymorphic types?

Summary: Are you expected to point to reasonable research directions? Influence of this work on other areas of Computer Science (data-flow architecture, verification methods for hardware,...) and other areas' influence (study of synchronous systems, complexity theory). What are we doing different from mathematicians?

Hope you find this useful,

Jay

Jay

jlb