Review of "On the Feasibility of Software Certification."

(Final Report, prepared by R.E.Keirstead for the National Science Foundation. SRI Project 2385, Stanford Research Institute, Menlo Park, California 94025, U.S.A.)

The report itself has seven chapters (29 pages) and is followed by three appendices (39, 14, and 47 pages respectively). It is well-written --motherhood statements are recommendably scarce!-- and is quite illuminating, as a picture of both the present state of the art and the American political/industrial scene. (It is this mixture that makes the report somewhat unbalanced: while the technical problems of software certification are certainly international, the discussion how a certification institute should get its revenue is more parochial

For the local political scene it contains a warning that should be repeate in this review: "Currently, there is some concern that formal or legal requirements for certified software may be imposed before the means for certification are available. Such requirements, without the technical means to accomplish certification, can only lead to disillusionement with certification, to the detriment of the entire software industry." Amen.

In stating our current inability to certify software the report is healthil explicit. Its analysis of the causes of this inability is, however, too superficial to justify fully its specific recommendations. The last appendix mentions the as yet unsurmountable difficulty in verifying sizeable programs "written in conventional programming languages with rich sets of primitives". After such a remark, one must make up one's mind: is there any hope that realistic verification techniques will become available that can cope successfully with such "rich sets of primitives"? The remainder of the report --and I am far from amaze gives very little support for that hope. Having identified conventional programm languages as one of the real culprits precluding verification, one could propose to exclude from the certification activity such programs that, on account of the way in which they have been written down, must be classified as "unverifiabl a suggestion that could do some much-needed harm to the popularity of these programming languages. The report --its authors seem more concerned with the certification institute being politically acceptable-- have not done so. In view of the political/industrial scene, this omission is not surprising. It is, nevertheless, depressing, for, as a result, much of the document deals with how to make the beast of a bad job.

* * *

The report shows a misunderstanding of the proper role of high-level programming languages, which may be at the root of many of our current problems. It is shown most clearly in: "The current state of formal proof [...] requires the proof to be developed at a representational level far removed from the pattern of bits that is the executable program in a real computer environment." (I have learned to become very suspicious when the word "real" is used in the above sense!) Clearly the author sees the pattern of bits as the programmer's final target, and the high-level language and its compiler as a software tool, as a means for generating that pattern of bits. But for a user, this is a very impractical interface. The semantics of his high-level programming language should be so well-defined, that he can totally disregard the compiler and the bit patterns it generates as for him irrelevant aspects of the implementation. Unless programmers learn to separate the definition of a programming language from its possible implementations, very few programs worth certifying will be written.

* * *

The hilarious suggestion:"The solution appears to be to extend the sequence that begins with machine language, procedure-oriented language, problem-oriented language. Increasingly higher levels of expressive language are needed plus far more exotic compilers to go back down the levels." is only quoted for my reader's amusement.

* * *

Finally, because "every program of consequence is probably incorrect" the principle author suggests "to consider other attributes of programs in determining certifiability". Is this courage or cowardice? Is this wisdom or folly? Will this be to the advantage or to the detriment of "the entire software industry"? I think I would prefer my readers to answer these questions for themselves, hopefully after they have read the report concerned and have an overview of the numerous and varied issues involved.

prof.dr.Edsger W.Dijkstra
Burroughs Research Fellow