WICHMANN84
Wichmann, B.A.; Is Ada too Big? A Designer Answers the Critics; CACM 27, 2 (February 1984) pp. 98-103.

Ada is undoubtedly one of the more controversial computer programming languages today. Many people have criticised it for being too large and/or complicated. This article, written by one of the members of the Ada design team (although not in any "official capacity"), attempts to address many of the criticisms.

As Wichmann is quick to point out, the language was ordered by the United States Department of Defence, and, since they seem satisfied with Ada, then by at least one criterion the language is a success. As is always the case when evaluating a computer language, the major design goals, and the intended area(s) of application, must be taken into account.

Much of the criticism related to Ada seems to implicitly assume that, in the words of economist E.F. Schumacher. "small is beautiful". However, is this an appropriate criterion for computer programming languages? The Department of Defence wanted a single language appropriate for a large applications area: this necessarily led to a large language. Having pointed out this fact, Wichmann addresses thirteen specific size-related criticisms, showing how (at least in his opinion) the criticisms are somewhat misguided or inappropriate. For example, to counter the claim that the language form would be easier to remember if Ada were smaller, Wichmann points out that Ada has only 62 reserved words, compared to 260 in COBOL. Similarly, while agreeing that it is easier to teach a smaller language, he contends that Ada makes it easier for academics to demonstrate such concepts as abstract data types, concurrency and error-handling, since they all exist within a single language.

The criticisms discussed above were, for the most part, rather generic in nature. Other criticisms tend to be more specific: they are proposals to remove specific features from Ada. Wichmann divides these proposals into three disjoint sets:

1) changes which (he feels) would be "disastrous to Ada"

2) changes which are "possible but of questionable value"

3) changes which are "practical and worthwhile"

Those features which he places into the first category (disastrous for Ada), he does so because he feels that such changes would "contravene the requirements of the language", turning Ada into a completely different language. These features include exception handling, concurrent programming and default initial values for types.

He concedes that some of the suggested eliminations would only "marginally change Ada", but contends that many are simply a matter of personal preference. This includes such features as fixed point facilities, more than a single exception, and separate compilation capabilities.

Finally, he identifies some proposed changes with which he agrees. Perhaps it is not surprising that this is the smallest category, containing only four specific features: unnamed array types, "when" conditions in exit statements, goto statements (although he points out that Ada has restricted the capability of the goto from that of Pascal), and entry families.

Wichmann concludes by agreeing that it is very possible to make Ada simpler by reducing its facilities. However, he questions whether the resulting language will be as useful to the intended user community. He points out that, even if every one of the "possible" changes were made, the resulting compiler would only ne twenty per cent smaller. If someone wants just a subset of Ada, that is all that they need use: no major advantage is to be gained by eliminating those features which that particular user does not require.

While the article is undoubtedly somewhat biased, it does seem to be well thought out, and intelligently written. Some of the "justifications" may lack substance, but he does not seem to deliberately mislead nor distort issues. The design of such an ambitious language must have been a very trying exercise. As Wichmann says, "in ten years, we will know how Ada should have been designed. It is the decisions we made in the original design phase and the alterations we make today, though, that really count."