Judging "HOS" from a distance

"HOS" stands for "Higher Order Software", originally the name of a methodology, now the name of a company. I know the company's product from four publications: [0] which introduced in 1976 HOS as "a methodology for defining software", [1] which appeared after the foundation of HOS Inc., [2] which oversells the product, and [3] which for all its brevity is very illuminating. Here is what I reconstructed from these sources.

Margaret Hamilton and Saydeam Zeldin (now President and Vice-President respectively of HOS Inc.) joined The Charles Stark Draper Laboratory in the mid 60's. There they got heavily involved in the programming of the Apollo on-board software and the like. In the course of the 70's they show symptoms of proposing a remedy for the problems encountered. They propose some sort of applicative programming language (which obliges its user to introduce a very elaborate nomenclature for intermediate results). Their main motivations seem to have been the following (i) a number of undesirable properties —such as usage of uninitialized variable— can be absent from the target code by virtue of the compiler's construction; (ii) the compiler can be constructed so as to produce only code satisfying a number of restart requirements (iii) the decision how the workload should be divided over a number of (quasi-)concurrent processes can be postponed until the compilation phase.

It is, in fact, [3] that makes clear that they propose to translate applicative programs into imperative programming languages; this was not clear from [0] and [1], which are incredibly poorly written. By way of illustration I quote from [0]:

> "We envision a scheme in which the definition of a given system can be described with an HOS specification language which, by its very nature, enforces the axioms with the use of each construct."

[(i) what is meant by "a scheme"?
(ii) how has the "given system" been given?
(iii) if we can "describe a definition", what is the (profound?) distinction between a definition and its description?
(iv) what means "enforcing an axiom"?]

> "The Structuring Executive Analyzer is a lower virtual layer module with respect to a given hierarchical HOS system in its dynamic state."

and from [1]:

> "A system development process is a system that develops another system. Such a system can be viewed as a process [etc.]"

The more technical parts of these papers are mathematically poor and notationally clumsy. The overwhelming impression is that the authors have had much more experience than education. But what kind of experience? I am perfectly willing to believe

that they have been able to improve upon what they refer to as "conventional techniques", but in some of their examples these are so backward that their achievement becomes rather unconvincing. I quote from [1]:

> "we discovered that many interface errors took place in the implementation when programmers would use instructions in an unstructured language, such as "GOTO +3". Errors would creep in when someone would come along, often the same programmer, and inadvertently insert a card between the GOTO instruction and the location at which it should have gone. Once we discovered the amount of errors that resulted, we enforced by standardization the use of instructions such as "GOTO A" rather than "GOTO +3". As a result, such errors never happened again."

[I can believe that! But admitting instructions such as "GOTO +3" was well-identified as a silly mistake more than 25 years ago!]

Several enthusiastic references to automatic flowcharters don't enhance the authors' credibility either; it makes one wonder how many of their development "tools" are, in fact, mechanized aids for the application of obsolete techniques.

* * *

Of the publications studied, [2] is the most bulky – more than 400 pages! –. It has about as

much technical content as a tract of Jehova's Witnesses. I quote by way of illustration — don't try to understand it — from [2]:

> "The approach described in this report is different. Instead of applying ad hoc programming constructs, it applies only constructs which are built with mathematical axioms and proofs of correctness. A library of provably correct operations is built. The operations manipulate precisely defined data types by means of provably correct control structures. [...] Most important, the mathematics is hidden from the typical user so that the method is easy to use."

James Martin wisely states nowhere what is meant by the magic formula "provably correct". He only states what it does not mean —[2], p3—:

> "When we say "provably correct" we do not mean that the program is necessarily without fault. [...] However the majority of bugs in programs today are caused by the mechanics of programming, inconsistent data, sequence errors and so on."

[The "mechanics of programming" is the culprit, so let us remove the need of "programming" by calling it "defining software".]

From [2] we learn that in the mean time HOS Inc. has "perfected" its product, not by cleaning

up the conceptual mess with which it started – [0] talks about "violation of this theorem": how does one violate a theorem? –, but by developing..... a graphics editor! And that is precisely the kind of would-be embellishment the professional software designer is only too happy to ignore. I am afraid that HOS Inc. has done some "market research" and has concluded that the incompetent ones form a bigger market.

Finally, by admitting James Martin to the board of their company, Hamilton and Zeldin have lost all credibility.

[0] Hamilton, Margaret and Zeldin, Saydean, "Higher Order Software – A Methodology for Defining Software.", IEEE Trans. on Software Engineering Vol. SE-2, Nr.1 (March 1976), 9–32.

[1] Hamilton, M. and Zeldin, S. "The Relationship Between Design and Verification.", Journal of Systems and Software, Vol.1, Nr.1 (1979), 29-56.

[2] Martin, James, "Program Design which Is Provably Correct.", Savant Research Studies, Carnforth, U.K. (1982)

[3] Cushing, Steve, letter to ACM Forum, Comm. ACM, Vol.25, Nr.12 (Dec. 1982), 951.

Plataanstraat 5
5671 AL NUENEN
The Netherlands

12 April 1983
prof. dr. Edsger W. Dijkstra
Burroughs Research Fellow.