The art of programming vs. software mass production

In the recent article “Software Engineering: The Turning Point” (Computer, September 1978), much weight was given to the idea of developing a “software inventory roughly analogous to the inventory of prefabricated circuits available from semiconductor manufacturers,” and to the consequent conversion of the “individual craftsman” programmer to a form of cut-and-paste systems engineer. While this is exactly what happened to the “individual craftsman” transistor circuit engineer in the time span from 1960-1975, there are several significant differences between software modules and hardware modules which will make this unlikely for programmers for a long time. The differences fall into three categories: (1) complexity of function, (2) quality of product, and (3) monetary implications of mass production. The implications that these differences have are as follows.

Complexity. The complexity of very simple salable software modules is somewhere in the range of 100-10,000 times more complex than that of salable semiconductor circuits. The documentation and the time to understand a purchased software module are roughly proportional to its complexity, especially if it must be made to do something just a little different. This causes a very expensive support burden to be placed on the producer of such software. The software producers almost never provide the quality of support which is taken for granted in the hardware side of industry, with the obvious result in a user’s make-or-buy decision.

Quality of product. When one is dealing with low-level-of-complexity integrated circuits they nearly always do what the specification sheet says. Some recent VLSI circuits are showing software-like surprises. But with software, we’re surprised when it does exactly what the specification book says. The make-or-buy decision is biased in this case by the desire to fix one’s own mistakes, rather than someone else’s.

Monetary implications of mass production. In the microcircuit hardware world, one thinks of selling reproductions of a single design between 100,000 and 10,000,000 times during the design’s lifetime. This results in the per-unit cost being very low compared to what an individual user would be forced to pay to make (for instance) a flip-flop in his own shop. Economics forces the user to buy the standard flip-flop.

But in the software market, a module costing 10-1000 times more to develop than an integrated circuit may sell several hundred or several thousand copies, while incurring proportional support costs. The result, except for a few very general-purpose software packages, is high cost to the user. Since the user in a large percentage of cases wants a “simplified version,” or a restricted case version, or a faster executing version, the make-or-buy decision is again tipped toward “We’ll do that in our own shop.”

One further factor weighs against the widespread use of standard software modules and it relates to a social filter effect. As people are exposed to software related professions it occurs to the individual craftsman types among them that software is the place they should be. The software industry, as a result, is loaded with proud individual craftsmen!

James M. Loe, Motorola, Inc., Government Electronics Division

November 1978