"Any clod can have the facts, but having opinions is an art."

Charles McCabe,
San Francisco *Chronicle*

THE OPEN CHANNEL

The Open Channel is exactly what the name implies: a forum for the free exchange of technical ideas. Try to hold your contributions to one page maximum in the final magazine format (about 1000 words).

We'll accept anything (short of libel or obscenity) so long as it's submitted by a member of the Computer Society. If it's really bizarre we may require you to get another member to cosponsor your item.

Send everything to Jim Haynes, Applied Sciences, UC Santa Cruz, CA 95064.

Prototype Programs

W. P. Dodd
University of Birmingham
United Kingdom

One of the panel sessions at the recent Fourth International Conference on Software Engineering was devoted to the topic "Concept Modeling." It was concerned with how the software engineer can determine exactly what the customer requires, and the discussion hinged around the importance of maintaining a dialog between the implementors and the customer. The need for this dialog was emphasized even for those cases in which there is no actual customer, and it was suggested that in such cases there should be a simulated discussion with a hypothetical customer.

The panel members seemed, however, to restrict these dialogs to the period up to the point of defining the task specification; thereafter the customer is, presumably, to live with whatever is produced. The panel considered this situation analogous to the dialog between the would-be purchaser of a new house and the selected architect; here also there are difficulties in communicating concepts, and, despite all the plans and sketches, once the building has commenced, there is very little the customer can do to affect the final structure. This may be a good analogy for current practice, but such practice is an inefficient and restrictive approach to software design. Instead, I would like to make a plea for the concept of prototype programs, which was mentioned by the panel but was dismissed on the grounds of cost.

In addition to considering that the construction of prototypes is an activity that we should be performing in any case, I wonder if we are already producing them without really admitting it. This idea is suggested to me by the frequently made statement that most (two-thirds) programmers nowadays are employed on "program maintenance." But not many analysts/programmers seem to enjoy being known as software maintenance engineers, unless perhaps they get called in over the weekend on quadruple pay—that doesn't happen very often. The fact is that not many of them are actually involved in maintenance; the large majority work primarily in program or documentation enhancement. Now surely this is very much akin to the refinement of a prototype system until it actually meets the customer's requirements. Such an approach is necessary because, despite all our efforts at setting up a dialog, the customer and the engineer still speak somewhat different languages.

If we set out to build an operational prototype rather than the production version first time out, we would have the opportunity to maintain the customer-designer dialog after the initial agreement on the specification. This approach has the advantage that the customer cannot say the system is not correct just because the prototype doesn't match his requirements. Instead he can point out what changes of the prototype he would like in the production version and do so with a great deal more precision because there is something concrete on which to base the discussion. The customer would acquire more confidence in the software engineer, and a large number of "maintenance" programmers would obtain a justified rise in status and self-esteem, since they would be working on the production version of a system rather than maintaining one that is inadequate. There might even be an increase in productivity, since engineers no longer would face either the pressure of getting the system right the first time or the disappointment of its rejection as inadequate.

As to the argument against prototypes on the basis of costs, if the above analysis is correct, we are probably already paying the price and more in hidden ways. In any case, why should we complacently assume we don't need prototypes when more established branches of engineering, aeronautical engineering for example, wouldn't dream of not producing a prototype?

The prototype concept may not be tenable in all application areas, but, even where there is no identifiable customer, it may be more fruitful to demonstrate a prototype version to potential purchasers, thereby exhibiting flexibility rather than a take-it-or-leave-it stance.

To return to the analogy of the architect-built house, obviously not many people can afford a prototype house, and not many houses, having been built, can be rearranged, have their orientation adjusted in some way, or easily have rooms added on. But it is just this sort of modification we are performing when we customize a prototype program.

However, one aspect of the architect analogy may work—the buyer's opportunity to view similar houses designed by the same architect. He can compare them with the architect's specification description and drawings. This comparison provides examples for the dialog between architect and buyer. In much the same manner, a prototype program can serve as the concrete basis of the dialog between engineer and customer. ■

0018-9162/80/0200-0081$00.75