

TECHNICAL REVIEWS:

THE ELIMINATION OF SURPRISES

By Daniel P. Freedman

Those of us involved in data processing are accustomed to a life full of surprises. We are constantly amazed at hardware innovations. We are often astounded by new software products; and unfortunately, we are often surprised by the appearance of errors in what we thought were stable, correct systems. Many of these surprises result from defects that have remained in the software undetected through the testing and production phases. Some of these surprises are errors that have been introduced during system modification. Regardless of the origin of these errors, the results are the same: Frustration, occasionally panic, always expense.

The exact cost of error removal is impossible to assess, but a number of traditional estimates are prevalent throughout the industry. One such figure is that 40% of system development costs are spent on defect removal¹. This is by far the largest figure in any category of system development.

A popular rule of thumb is that the cost of the removal of an error increases by an order magnitude at each subsequent stage of systems development. That is, if an error costs one unit to remove at the requirements definition stage, it will cost ten units at the functional specification stage and one hundred units at the high level design stage. From there the numbers keep increasing. These figures apply to software products; that is, systems which are sold to an external customer. Even if the figures for in-house products were 50% less, error removal would still be a costly endeavor.

Another series of figures has been stated for avionics systems. These costs, as noted by B. Boehm, address the cost of creating a line of code versus modifying that line once it is part of a finished system. According to Dr. Boehm, a line of code costs approximately \$75 to produce and approximately \$4,000 to modify². Awesome figures!

The point is clear! The earlier an error is detected, the less expensive its correction. This axiom, of course, applies not only to the software industry. Imagine please -- you are having a house built; and while reviewing the plans in relation to the house site, you notice the view would be greatly improved by shifting the structure four feet to the west. All things being equal (land ownership, correct terrain, etc.), the cost of the change at this phase is very close to zero.

Contrast this with the situation where you find a completed structure. Here, too, the view could be greatly improved by shifting the structure four feet to the west. This would be an extremely costly change, if not an impossible one to make.

These costs are limited to the expense of actually fixing an error. They are not the costs associated with the results of the error. Resulting costs could range from minimal and trivial, a simple program restart, to disastrous, a major airline accident over a major city.

Of course, errors should be caught in testing! That's why we spend 40% of our development cycle on unit and system testing. The Quality Assurance Departments should catch the errors that elude the test process. The unfortunate reality is that testing procedures and Quality Assurance do not catch all the errors.

There are theoretical and pragmatic reasons why systems testing cannot possibly detect all errors in a system. The combinatorics of testing all possible paths are too tremendous. In large systems it is impractical, if not impossible, to exhaustively test every possible path. If testing is your only method of validating a system, then it is impossible to validate an untested line. If it is impossible to test every line, then it is impossible to validate a system. But this is an unacceptable reality. There must be another alternative!

Quality Assurance and Auditing Departments are another protection against the releasing of defective systems. Again, it is impossible for Quality Assurance to bear the responsibility for assuring the absolute correctness of a software system. It is true that Quality Assurance can assure that certain standards and procedures have been followed. Quality Assurance can certify that certain "levels of acceptability" have been reached, but certain "levels of acceptability" is not the same as correct.

Another process is required to supplement the testing and quality assurance procedures for program validation and verification. This process, which is called Formal Technical Reviews, is the detailed examination of a finished product by technically competent individuals. The main purpose of such reviews is to assure the technical quality and content of a finished product. Competent people read every line of specification, view every diagram of design, examine every line-executable code. After this in-depth examination, these people certify that to the best of their understanding and based upon their relative technical competence, the product is correct.

The review process is not absolutely perfect. If it were, it could replace Testing and Quality Assurance, which is clearly not the case. Rather, Technical Reviews are based on the premise established earlier in this presentation: The earlier an error is detected, the less expensive it is to remove that error.

Technical Reviews examine each product or subproduct involved in a systems development life cycle. The only necessary commonalities among products scrutinized by the review process are that the products be considered finished by the producer or producers and that the products are part of the required steps in the systems development life cycle.

There are many forms of Formal Technical Reviews. Each form has advantages and disadvantages. However, all forms of Formal Technical Reviews result in a better technical product. One of the most well known of Formal Technical Reviews is a Walkthrough. In this form of Review, the producer (or some other knowledgeable person) leads the reviewers through the product. The reviews mentally simulate the logical process, code, or system requirements and attempt to find errors or potential problems with the product. As in the case of all Review processes, it is also advantageous if the reviewers mention the positive aspects of the product.

An Inspection is another form of Technical Review. In this format, the producers work from a preconstructed checklist. Just as an aircraft pilot prior to take-off inspects the status of the equipment, the reviewers inspect the status of the product. Just as the pilot verifies the inspection based on a predeveloped checklist, the reviewer also verifies the product based on a predeveloped checklist. If the checklist is carefully constructed and followed, there are rarely errors of omission. The checklist guides both the pilot and the reviewer.

The Independent Product Analysis is the final format that a reviewer may select. The reviewer examines the product unguided by either the producer or a checklist. The reviewer is neither guided by the knowledge of the producer nor by the preconceived images of the checklist.

Regardless of the form of Technical Review, it has been shown to be cost effective and worthwhile since it allows errors or potential problems to be detected earlier in the systems development life cycle. It is difficult to quantify the exact cost effectiveness of Technical Reviews for a number of reasons. The primary problem is that, to the best of our knowledge, there are no accurate baselines as to the actual cost of systems development.

Second, in my client organizations where Technical Reviews are regularly being conducted, there is no analytic method of cost effectiveness being employed. However, without exception, client organizations feel that the Review expenditure is worthwhile. They base this conclusion on quasianalytical grounds: The people who participate in the reviews report that they have found major errors which if found later would have been more costly to correct than when found at the time of review.

The Quality Assurance people report that there are fewer products being rejected for failing to meet the Quality Assurance criteria. They further report that individual products are requiring less time in the Quality Assurance process. Apparently, the products are being delivered to Quality Assurance in a more secure form. The format is clean and concise. The code is readable and well documented. The diagramic material is clear and represented in a consistent manner. The English language sections are grammatically correct and unambiguous. All in all, the product quality is more uniform; and uniformity and consistency are the first prerequisites to clarity.

Other intuitive evidence comes from post-implementation specialists (maintenance people). A number of my clients have specialized maintenance areas. These individuals are truly post-implementation specialists. Once a product has been approved by Quality Assurance, the product is no longer the responsibility of the development people; rather the responsibility shifts to the maintenance specialists. These individuals report that there are fewer surprises occurring in reviewed products; and when these errors do occur, they are easier to trace. This traceability is not surprising. A by-product of the review process is that the system must be built in small, manageable units. In order for a product to be reviewable, it must be decomposed into comprehensible units.

Normally, a reviewer has no more than two hours to review a product. Therefore, a unit of separable work can be no larger than an amount that a highly talented and skilled randomized group of individuals can comprehend with no more than two hours preparation. Most maintenance specialists will tell you that an error takes five minutes to fix -- after it is found. By restricting the "working" product to a unit requiring no more than two hours to comprehend, the scope of a potential product has been, in its construction, limited.

Obviously, this fact cannot apply to all systems. Highly integrated and tightly coupled systems³ may not be easily partitioned into small reviewable units. But this lack of ability to isolate reviewable parts is more a function of the design of these systems than a comment on the review process itself.

In the general case, a systematic review process results in the architecture of reviewable systems. Designing systems in reviewable units results in the creation of loosely coupled, highly cohesive designs. According to local parlance (structured methodologies), systems designed according to these principles ought to be cost effective.

Organizationally, reviews may demonstrate that products are often released based on business and political considerations rather than technically valid considerations. However, the ongoing regular use of Formal Technical Reviews will greatly improve the quality of product generated by an organization. This will hopefully encourage those in decision- and policy-making positions to give more emphasis to the technically valid considerations. As this begins to occur, the organization will realize more profit as a result of internal efficiency.

¹ Jones, Caper. ITT, 1980

² Boehm, B.W. "Software Engineering: R&D Trends and Defense Needs, "Research Directions in Software Technology, Cambridge: M.I.T. Press, 1979.

³ Yourdon, Ed and Larry Constantine. Structured Design, New York: Yourdon, Inc., 1975