# *Editor's Comments*
# Re-engineering the Organization

The notion of re-engineering the organization through information technology has roots going back quite a few years, but it has recently gained considerable new attention. This interest arises from the growing realization that substantial changes must be made in most current applications if an organization hopes to achieve strategic benefits from its information system. It also comes from advances in technology that make such an approach increasingly feasible. In the July-August 1990 issue of the *Harvard Business Review,* Michael Hammer states the case for the doctrine in his excellent article, "Reengineering Work: Don't Automate, Obliterate."

Re-engineering should be an important goal in any effort to develop an IS plan for an organization. Drastic changes have taken place in information technology over the life of most existing information systems. Even if we suppose that an application developed a dozen years ago achieved the right design tradeoffs at the time its requirements were specified, the design is almost sure to fall short of today's needs. The power and cost of the technology have improved in some cases by a factor of 10 or more, making it entirely feasible to provide functionality and integration that were out of the question when the initial requirements were established. Continued "maintenance" of the application over the intervening years may have partially corrected for shifts in the environment, but such changes tend to be quite local and minor compared to the underlying opportunities for improvement.

Symptoms of obsolete design concepts are manifested throughout almost any large application developed a decade or more ago—in its failure to satisfy cross-functional needs, its limited functional capabilities, its inability to provide selected and tailored information drawn from widespread parts of the organization, its inflexibility in meeting changing needs, and its unfriendly interfaces.

Many of these weaknesses stem from the limitations of the conventional third-generation development process available when the systems were put together. The process suffered from an intrinsic flaw. It assumed that the organization—management, designers, *somebody*—could specify the right up-front "requirements" for an application, leaving to the technical staff the task of implementing the system as specified. Organizations soon learned that they had to establish tight discipline over the process to have any hope of success. Heavy emphasis was placed on getting the requirements right, and then keeping the specifications as stable as possible over the long development cycle. Not surprisingly, delivered systems often failed to match user needs and expectations. Despite vast sums spent to correct errors of omission and commission in the specifications, it usually proves extremely difficult to correct fundamental structural flaws in a system.

We still implement most of our heavy-duty, mission-critical systems through such a process. It's no longer good enough. We need to make some fundamental changes in how we go about developing information systems.

For starters, we need to do a better job at specifying requirements. The process often used to elicit requirements tends to constrain users from thinking anew about how information technology might better serve their needs; it is more likely to motivate pedestrian improvements in existing business practices. Jim Wetherbe's article in this issue provides some valuable suggestions about how we might improve the process of requirments specification.

It's difficult to argue that we should not do all we can to get sound initial specifications, but we need to do more than that; we need to employ a development methodology that makes it relatively cheap to be wrong in our specifications. Try as we may, we will *never* get a completely satisfactory statement of requirements. The world changes, the technology changes, and—most of all—users' perceptions change. Technical management often interprets the changes in users' perceptions as evidence that managers do not know their own jobs. In fact, however, we should welcome and encourage such change, because it is a symptom of organizational learning. If we inhibit these changes, we will stifle adaptation of the information system and the organization. It is only through such adaptation that we can ever hope to develop a successful system and keep it responsive to changing needs.

This brings us back to the question of re-engineering the organization. By its very nature, re-engineering calls for entirely new ways of carrying on the activities of the organization. Re-engineering forces stakeholders to deal with some difficult abstractions: they must rethink the fundamentals of the business, unconstrained by the existing way of doing business. The bigger the leap from current practice, the more difficult it is to accomplish and the less likely stakeholders will be able to get it right the first time. If the development methodology assumes they can get it right the first time, and then imposes a huge penalty if they fail (which they will surely do), then prudence calls for timidity in striking out in new directions. If, in contrast, new methods can be implemented quickly and cheaply and then can be corrected as the organization learns, we stand a much better chance of exploiting opportunities presented by information technology.

A key need, then, is an adaptive development methodology. This, in turn, is achieved with a highly productive development environment that allows the developers to begin with a good approximation of where they are headed, then lets them modify the system through an iterative process of feedback from users. It becomes feasible to implement a large system in bite-sized chunks, using small teams able to bring a new application into production over a short timespan.

Some leading organizations are beginning to put together the kind of productive environment that allows this evolutionary approach to re-engineering the organization. It is not ureasonable to aspire to a tenfold improvement in productivity through a combination of a sound enterprise architecture, first-rate development tools, small teams of very competent analysts, the avoidance of unnecessary red tape and bureaucracy, and a strong partnership with users. IS managers appear to be increasingly aware of the importance of a productive and adaptive infrastructure of this sort and are willing to invest resources to achieve it.

Needless to say, if our goal is truly to re-engineer the organization, we will also need vigorous and continual support from top management. Gaining such support may be the hardest part of all.

**James C. Emery**