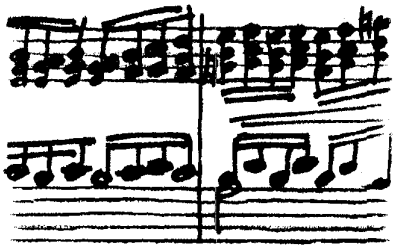


Large-Scale Project Management *Is* Risk Management

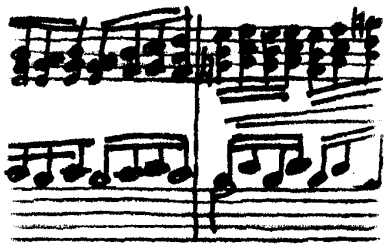
ROBERT N. CHARETTE, *ITABHI Corporation*

Because large-scale software projects increasingly affect the public good, the "normal science" paradigm is proving insufficient to model their complexity and potential consequences. The "postnormal science" paradigm offers a better fit, using a robust management approach predicated on a risk-taking ethic.

Two years after the start of the US Federal Aviation Administration's $4.3 billion Advanced Automation System project contract in 1990, the Government Accounting Office stated that continuing delays in the deployment of the Initial Sector Suite System, a key component of the AAS, could "have the potential for affecting FAA's ability to handle safely the predicted increases in traffic into the next century."[1] Later that year, the AAS project schedule was extended by 19 months. The FAA blamed the delay on their underestimating the development and testing time for the ISSS software, as well as on unresolved differences in the system specifications caused by changes to the requirements.

By April of 1994—following an additional 14-month schedule delay in early 1993 (blamed once again on ISSS-related software problems)—FAA management declared the AAS project "out of control." At that point, the cost for AAS completion was predicted to reach over $7 billion, with yet another schedule slip of up to 31 months possible. At this point, the FAA effectively suspended the AAS program.

0740-7459/96/$05.00 © 1996 IEEE



The FAA recently announced that a reduced-functionality ISSS under a restructured and curtailed AAS program called the Display System Replacement project will become operational in 1998—six years late, fourteen years after the project started, and seventeen years after it was initially defined. The current estimated cost-to-complete for the DSR project totals $5.6 billion.

Until that time, and assuming no further setbacks, the FAA and the flying public can only hope that the current air traffic control system, with its over taxed and breakdown-beset 1960's era computer systems, will not become the source of a catastrophic accident.

BEYOND NORMAL SCIENCE

Large-scale software projects like the AAS or the US Defense Department's Strategic Defense Initiative—which were estimated to require more than 2.3 and 10 million code statements respectively—are characterized by high decision stakes and high levels of system uncertainty. As such, they are poor candidates for being planned or developed under the prevalent "normal science" model of project management.

Normal science assumes that large-scale software projects are like puzzles to be solved: using reasoned trial and error, based on accepted engineering paradigms, the pieces will fall in place. However, as Frederick Brooks has pointed out, creating software systems is one of the most complex tasks ever undertaken. No matter how we might otherwise pretend, for large-scale software projects there are no edge pieces to guide us, no picture on the box we can refer to when stuck, and no assurance that we have all the pieces or even that they fit together.

Increasingly, large-scale projects most closely fit into what is called "postnormal science." They have objectives that are unprecedented in breadth and depth—such as reliability requirements of 99.99999 percent in the AAS. Ambiguity, continuous change, and complicated feedback loops dominate our project-management decision making. Because our endeavors are often unique, no one has the requisite expertise we need for planning and implementation. Project success or failure affects the public directly and indirectly, often creating unintended socioeconomic impacts. As they continue to increase in size, complexity, and the potential for ill effects if they fail, more and more of our projects will fall into the "postnormal" category.

The implications for project management are two fold. First, many of the assumptions underpinning traditional project management are tenuous at best and incorrect at worst. A combination of change, complexity, discontinuities, diseconomies of scale, nonlinearities, and the consequences of failure serve to undermine them. Second, project management should—I believe, *must*—be propelled foremost by a process and philosophy of risk management; it should be the central actor in project management instead of just another member of the supporting cast.

NORMAL VERSUS POSTNORMAL SCIENCE

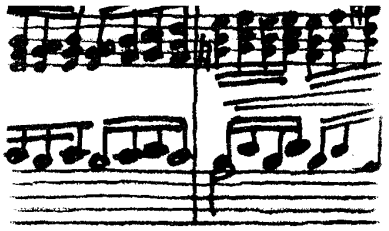
In 1962, scientific historian and philosopher Thomas Kuhn published a breakthrough book, *The Structure of Scientific Revolutions*. In it, Kuhn wrote that scientific paradigms, what he called "normal science," are "accepted examples of actual scientific practice—examples of which include law, theory, application, and instrumentation together—[that] provide models from which spring particular coherent traditions of scientific research."[2] Scientists and engineers operate under these accepted conceptual world views or para- digms to attack important contemporary problems and to define legitimate areas of research.

Scientific progress, Kuhn asserts, takes place by a continuous process of "puzzle-solving," that is, scientists use the accepted paradigm to solve current questions of interest as well as to resolve paradigm–nature anomalies—areas where the paradigm doesn't seem to quite fit reality.

Normal science assumes that large-scale software projects are like puzzles to be solved.

Occasionally, an anomaly is of such significance that it directly calls into question the whole paradigm. The scientific community then shifts from unanimity into crisis, conflict, and turmoil until a new paradigm emerges to explain the anomaly. Things then run relatively smoothly until another crisis-causing anomaly is discovered and the process repeats itself once again. Thus, Kuhn contends, scientific progress is a matter of relatively peaceful periods of paradigm refinement punctuated by intense periods of dynamic change. Our move from an Aristotelian to an Einsteinian concept of physical laws, for example, required several such transformations.

Changing times. In the mid-1980s, other scientific historians and philosophers, notably Silvio Funtowicz and Jerry Ravetz, began to argue that Kuhn's view of scientific progress was too narrow.[3] Funtowicz and Ravetz contend that while Kuhn's model may have been representative of past scientific progress, it is no longer entirely applicable. They claim that "normal science" presumes an insular community of scientific interest from which the public is generally excluded; discussion of what is beneficial scientific advance-



ment is limited to the scientists themselves. This view implies that there is a scientific answer to pressing societal problems, with scientists—because of their objectivity and expertise—ideally positioned to address them.

Not all hypotheses can be tested by experimental means.

Funtowicz and Ravetz point out that today, however, scientists are being asked to reach solutions for complex *public* problems such as genetic re-engineering, global warming, and the like. With these types of problems

- the facts of the matter are uncertain,
- the values involved are in dispute,
- decision stakes are high and decisions urgently needed, and
- the public is deeply involved in making decisions and deeply affected by those that are reached.

Funtowicz and Ravetz maintain that the normal-science model, with its practice of using "hard" scientific inputs to make "soft" policy decisions, is not prepared to resolve these new, complex types of questions where "hard" policy decisions have to be made using "soft" scientific inputs. They go on to argue that normal science's tradition of being objective and ethically neutral does not fit many of the questions being asked of it today. An example? "Are the implications of human gene re-engineering, such as politically motivated eugenics, value free or ethically neutral?" The skills scientists require to fully address complex questions like this are usually beyond standard training. Scientists performing human-genetics research are not trained as ethicists nor are medical insurance actuaries trained as public-health policymakers. Still, each area is affected by what these scientists discover.

Funtowicz and Ravetz assert that science and its practice cannot be automatically separated from how it is used, above all when the consequences can greatly affect the public good. They further assert that even paradigm change à la Kuhn is not sufficient, nor will it ever be sufficient, to solve the messy technology-cum-public issues now upon us.

Paradigm departure. To deal with the pressing scientific problems of today, Funtowicz and Ravetz propose a trans-science model they call "postnormal science." This new model does not imply that the scientific method is invalid, but rather that certain questions or problems now being posed do not fit the scientific process; not all hypotheses can be tested by experimental means. A major argument brought against SDI, for example, was that the system could not be "fully tested" except in actual use.[4] As a result, the confidence that it would work when needed could never be attained. The implications of deploying such a system are not difficult to imagine.

Likewise, postnormal science does not imply that the normal-science model is irrelevant, but only that it is most germane when *both* problem uncertainty and decision stakes are low. Once either factor starts to increase, the usefulness of the normal-science approach decreases rapidly; solutions will be based upon group judgment rather than objective "facts." Normal science, therefore, should be viewed as a subset of postnormal science—applicable to many (if not most) problems, but certainly not to all.

PROJECT MANAGEMENT AND NORMAL SCIENCE

Contemporary project management has been heavily influenced by normal science over the course of its theoretical and practical development. Beginning at the turn of the century, in a period of intense scientific achievement, several management theorists and practitioners believed work could be improved by the application of scientific methods.[5] Notable among them was Frederick W. Taylor, who codified this belief into four principles of management, published in *Principles of Scientific Management*. According to Taylor, management should

(1) replace individual workers' rule-of-thumb methods with specialized, scientifically developed approaches;

(2) select, train, and develop workers using scientific methods, so each worker performs the right job;

(3) bring together scientifically selected workers and scientifically developed work to gain the optimal results; and

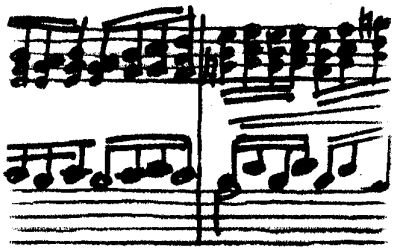
(4) ensure an equal division of work and responsibility between management and workers, and foster close cooperation between the two.

Project-based structure. Many people later refined Taylor's principles, including organizational theorist Luther Gulick. In 1937, Gulick presented the idea that a project-type organizational structure—one responsible for achieving a specific organizational purpose—could be more effective than the more traditional functional forms advocated by Taylor and others. Gulick's idea seemed to be supported by the numerous successful scientific and engineering efforts of World War II—particularly that of the Manhattan Project.

The success of difficult or "impossibly large" projects during the war years helped spawn further refinements of the project concept, culminating in the archetypes of modern project-management theory and practice: the 1950's era US Navy's Polaris Missile Special Projects Office and, for software particularly, the SAGE air defense system.[6]

With so many years of success in applying Gulick's concept behind us, we tend to passively take for granted the premises underlying project management. However, even a cursory glance at those premises highlights the influence of normal science.

- A project is defined as a clear-cut



investment activity with an explicit purpose and a distinct beginning, duration, and end.

- A project represents the lowest opportunity cost: It is the most beneficial option for expending scarce resources.
- At least one solution exists given the project's purpose, meaning that the project is *feasible* (it can be technically accomplished), *suitable* (it can be managerially accomplished), and *acceptable* (the project's purpose can be achieved).
- The time and resources required can be accurately predicted.
- The environmental context is well-understood and fixed, and "success" can be defined and measured.
- The risks involved, including their worst-case consequences, can be contained.
- Failure to meet the project's objective is caused by a lack of proper skills or their employment, rather than because the project is infeasible, unsuitable, or unacceptable.

This description is admittedly ideal. On even the most well-formed project, you would be hard pressed to completely meet any of the characteristics, let alone all of them. The description also contains a presupposition: That you have not only sufficient information to define a project, but that the information is accurate enough to let you predict future events. However, even in small software projects, information can be in short supply and of suspect accuracy, and thus prediction becomes difficult. Insufficient and inaccurate information forms small cracks in the premises of your project's foundation. Usually, the cracks do not undermine its structural integrity and you can successfully field the project, although often slightly late or over-budget from your original predictions.

When your project is large and complex and the amount of information and its accuracy decline rapidly, prediction becomes more akin to fortune telling. Structural anomalies or fissures start to appear in each premise. Like water seeping through an earthquake-weakened dam, the cracks reveal themselves through project consequences: missed schedules, exceeded budgets, and delivered systems that don't operate correctly or at all. For large-scale software projects—50 to 65 percent of which are ultimately canceled—the flow through the cracks has become a torrent.[7]

PATCHING THE PARADIGM

According to Kuhn, when cracks appear in a paradigm steps are taken to patch them up, first through a series of refinements. Over the years, various refinements in software engineering's basic paradigm, the waterfall life-cycle model (which is itself a codification of earlier stagewise development models), have been suggested. Requirements traceability, design inspections, code reviews, configuration management, quality assurance, process improvement, abstraction, iteration, rapid development, and so on have all been added to what is now considered good if not essential software engineering practice, especially for large-scale software projects.

Efforts taken to shore up project management or the control aspect of software engineering's paradigm continue to receive the most attention, as it is almost universally perceived as the major fissure eroding project success. Over the last decade, risk management has increasingly been seen as a useful patching material to fill the project management fissure and strengthen its surrounding walls.

Risk first. Barry Boehm's spiral model was the first major endeavor to make risk management a formal software engineering activity, especially in large DoD software projects.[8] The spiral model sought to consolidate previously proposed process-model refinements—such as the evolutionary development and transform model—into a single, unifying meta-process model, as well as to make risk management a much more visible and important part of project management.

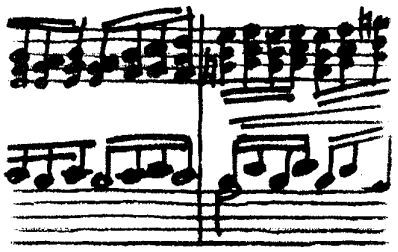
The spiral model uses a basic four-stage, cyclic, risk-driven decision process as a metaproject management control mechanism.

1. Determine project objectives, constraints, and so on.
2. Identify risks, evaluate alternative courses of action, and resolve risks in the course chosen.
3. Implement the selected course and verify its completion.
4. Determine whether the risks are at an acceptable level to proceed to the next decision stage.

The decision process overlays the individual phases of the meta-process model. Before a life-cycle phase is initiated, management must decide whether the project situation is currently acceptable, feasible, and suitable. The spiral model uses the level of risk exposure as a key decision metric for determining this. If you perceive project risks as being too great at any phase, you must reduce them before proceeding. For example, you might have to develop prototypes to reduce feasibility risk or select a special case of the meta-process model to lower suitability risk. Thus, in the spiral approach, how well you manage risk is the overriding factor in developing and managing software.

When your project is large, prediction becomes more akin to fortune telling.

Risk experience. The spiral model has been used by many companies, particularly in the aerospace industry, with



varying degrees of success. It was refined by The Software Productivity Consortium to strengthen its risk management and implementation aspects.[9]

At the start of a large project, accurate cost or schedule predictions are unlikely.

Still, few organizations have the requisite risk management expertise to apply the spiral model properly.[10] Generally, software project managers are not conversant with formal risk management, its relationship to decision making, nor how to integrate risk management into existing project management activities. The spiral model has increased the visibility of risk as an issue to be seriously considered in software project management, albeit not to the degree that might be wished.

To overcome the lack of risk management expertise in the software engineering community, several efforts have taken place since the spiral model first appeared.

- The Software Engineering Institute has an established software risk management program.
- The UK Government's Centre for Information Systems has created guidebooks on risk management for enterprise-, program-, and project-level managers in computing and telecommunications.
- The DoD, under its Software Acquisition Best Practices Initiative, developed a guide to acquiring best practices, designating formal risk management as a paramount practice.

Also, as the references at the end of this article show, numerous books and articles on the subject have also appeared in recent years.

SHATTERING THE PARADIGM

Despite efforts to improve its practice, risk management is still primarily regarded as an additional support activity of project management and not as project management's central tenet. More often than not, risk management is not considered at all because to do so "contradicts" a core premise underlying a software project. By definition, a project should be doable, not "risky"; except under exceptional circumstances, a project's chance for success should be significantly greater than the likelihood of failure. A project should also be well-defined, represent the lowest opportunity cost, and so on. After all, if it weren't, it wouldn't have been approved, right?

The risk stigma. Until they are completed or canceled, all projects have risk. However, for project managers to admit to "riskiness" can be seen as admitting to "not fully understanding the problem" or being "overly pessimist," or worse, "not a team player," among other things. This association of risk with something being wrong leads to cognitive dissonance: A belief is held in spite of evidence to the contrary. Many large-scale, extremely complex, and unprecedented DoD software projects, for example, are universally seen as risky yet are declared by fiat to be "low-risk" so as not to imperil future funding. Similar things happen in industry, and not only on large-scale software projects. Given this mindset, it is difficult to make a case for performing risk management.

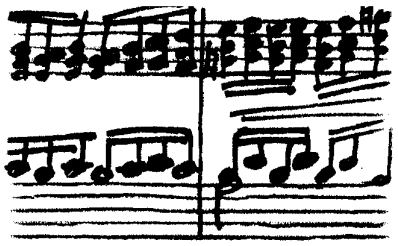
At best, risk management is viewed as one of those "self-evident" activities: software projects obviously involve risks that need to be managed. However, few project managers see any compelling reason or need to make risk management a separate, formal activity, let alone the quintessence of project management. This is unfortunate. In my experience, project success—particularly for large-scale software projects—is severely limited by this perspective.

As software projects become large-scale, the effects of complexity, ambiguity, change, and uncertainty dominate, acting like a confluence of storm-swollen rivers that rapidly undermine the premises that "normal" project management is built upon. That a project is feasible, suitable, and acceptable cannot automatically be presumed for large-scale software projects.

Cracking premises. Consider acceptability, for example. What does it mean in an AAS-like project? How can you clearly define investment objectives when the project spans multiple communities of interest with conflicting expectations and definitions of success? For a project of this sort, the primary community of interest includes not only the FAA customer, but the airlines, the companies that depend upon the airlines for routine business, and millions of passengers, not to mention Congress. Each of these groups is affected to varying degrees by the project's success or failure, and the consequences of the latter are much greater than merely the project's cost. Unintended consequences and attendant risks are also almost universally overlooked.

The acceptability premise is further weakened by the fact that large-scale projects are long-lived, and thus their objectives will assuredly change as economic conditions shift, technology improves, experience clarifies needs from desires, and so on. Each change in turn affects technical feasibility and managerial suitability as well. Accurately predicting cost or schedule at the start of such a project is highly unlikely, not only because of insufficient data, but because the premise of a fixed environmental context is violated as well.

Other assumptions are also eroding. In normal software projects, a definite end to the project is pre-



sumed. However, for large-scale software projects, with their long development and operational lives, their deep connection to a vast community of interests, and their tight coupling into huge programs made up of similar projects, this premise is severely eroded if not completely washed away. For example, the current DSR project is only one part of a much larger FAA modernization program that totals over $30 billion. Whatever happens with DSR has ripple effects well after it is officially delivered. Even DSR success will bring about changes in ways that are unanticipated. To paraphrase John Donne, no large-scale software project is an island, entire of itself.

Normal projects are assumed both to be beneficial and to represent the lowest opportunity cost option. For large-scale projects, neither assumption may hold. The realizable benefits of the FAA's air traffic control modernization program have been in dispute since its inception.[11] Furthermore, it is sometimes the case that not doing a large-scale project is better than trying and failing because of the way public perceptions are permanently shaped. As originally defined, SDI, and to a degree AAS, are cases in point.

LIVING IN A POSTNORMAL WORLD

With such strained or shattered "normal" project premises confronting us, we have two choices. We can continue to try to repair the normal-science paradigm of projects and project management and hope failure rates decline. Or we can decide on an alternative view.

Large-scale projects such as AAS or SDI fit the postnormal perspectives much better than the normal-science view because

- the "facts" of the situation are highly uncertain,
- project values and expectations are in constant dispute,
- decision stakes are very high,
- decisions are needed urgently, and
- whatever happens, a broad community of interest will be deeply affected.

Not all large-scale software projects meet these criteria, of course, as not all have a public face per se. It could be argued that most don't, at least not at the present time. However, this situation is likely to change. As software becomes ever more ubiquitous, increasing numbers of large-scale software projects are directly affecting our lives. Indirect impacts are also growing.

New criteria. The assumption that large-scale software projects are just like small ones, only bigger, must be abandoned. To effectively deal with postnormal-type software projects, project managers must adopt a different operating premise: Large-scale software projects belong to the postnormal world until proved otherwise.

However, scaling back to the "normal" project domain should not be seriously considered unless the following criteria are met:

- The project's risks (and rewards) are fully and completely understood by all parties potentially affected.
- The risks are thoroughly, continuously, and visibly assessed and managed.
- Project management provides the leadership to actively control the risks.

To ensure that the project's risks are fully and completely understood by the parties potentially affected, managers must openly define and share risk (and success). When they do, not only will all parties better understand the project's purpose and benefits, but when inevitable difficulties arise they are likely to respond with rational action rather than gut reaction. Each manager must also articulate the point of unacceptable risk, where the project is no longer beneficial,[12] the risks outweigh the rewards, or a project turns out not to be doable in its present form. To continue stubbornly past these points believing that something can be salvaged is not only folly, but invariably creates problems when you attempt the idea again in the future with a different approach.

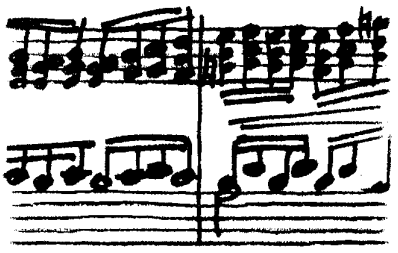
New ethic. We must develop an extra sensitivity to risk or, more appropriately, a *risk-taking ethic* [13] which holds that

- success entails taking on risks, sometimes very great risks, but doing so intelligently;
- risk is not something to be feared or avoided, but something we can profit from;
- change is not feared, but embraced; and
- by mastering the details, the risks can be mastered as well.

A postnormal project is filled with dilemmas to manage as opposed to problems to solve, and every decision has a potential for negative consequences. The high uncertainty and high decision stakes dictate that *every* decision, whether it be made by the project sponsor, project manager, the software team leader, or the individual programmer must be at a level of acceptable risk to the greatest possible degree. In other words, each decision must be assessed for risk.

Decision making. It is essential to implement a quality decision-making process that has risk as the overriding concern and that actively searches for risk in every decision, assesses risk to see if it is too great, and if it is, takes positive action to reduce it.[14] A primary characteristic of such a decision process is that it must be visible, repeatable, and measurable. This ensures that a base level of consistency

Large-scale software projects are postnormal until proved otherwise.



is achieved, helps others to understand why and how a decision was made as well as how to improve future decisions, and if needed, how a decision can be reversed with minimal damage.

Estimating the current situation is a powerful way to organize your thoughts and propose action.

To be most effective, the decision process must pragmatically support how project members make decisions involving risk on a daily basis. This implies that process cannot be bolted on as an afterthought, be ad hoc in nature, nor performed once a quarter or life-cycle phase. It must be embedded into everyday work practices. The process must also support different decision makers' perspectives and work contexts. It should help program managers, project managers, or programmers answer specific questions about financial, technical, legal, political, or any other risk or combination of risks involved in the decision making relevant to *their* work environment. Just as a rancher sees a cow differently than a microbiologist, a project manager sees a risk differently than a programmer.

It is imperative that you assess risks at a fine level of granularity so informed decisions can be made, but also at a level coarse enough that actions taken to manage them are efficiently implemented. Take, for example, two statements: "There is a timing risk in the system design" and "There is a 30-percent chance of the message buffer overflowing resulting in a 10-milliseconds delay." The former is useful to the project manager, the latter to a software team leader. Information concerning risks and implications of risk management actions must also flow freely across the project, as well as include multidisciplinary concerns of the stakeholders. Otherwise, incomplete or suboptimal decisions are the likely result.

In a postnormal environment, the most effective decision process accounts for risks from the initial project definition time until the project is retired. It does little good for a project plan to be developed and then checked for risk afterwards; risk must be the first input to any initial project definition and be continually reassessed throughout the project's development and operation.

Because decisions are so interlocked in large-scale projects, the consequences of a few or even one risky decision can be quickly multiplied a hundredfold to disastrous effect. Because our knowledge of a decision's effect is limited, even good knowledge of the separate risks involved can't prevent mistakes in an individual's intuitive judgment of the effects of managing those risks today. A postnormal project can only succeed when a project team has a risk-taking ethic and sees itself as operating in an enterprise situation on an undertaking of scope, complication, and most of all, risk.

Leadership. Even if the risks are fully and completely understood by all parties potentially affected, and the risks are thoroughly, continuously, and visibly assessed and managed, you still need a project management team that can provide leadership and actively control the risks. Without leadership, "risk-taking ethic" is just a meaningless phrase. Management must be proactive, or as Stephen Covey puts it, more than "merely taking the initiative," management must create a culture where risk is not synonymous with disaster.[15]

Being proactive means taking responsibility for the choices made on the project. It means spending the time necessary to understand how decisions are made and how they can be improved. It means looking at decisions as correct or incorrect, rather than right or wrong, and being able to reverse decisions that are incorrect. And it means management directs its energies at achieving success, not laying blame. Proactive, risk-taking project management stresses cooperation, collaboration, integration, and balance of action.

Paradigms are important to the way we see the world. They create a framework of thought for understanding and explaining reality. They also define the rules that we act under and what we consider acceptable behavior. If the paradigm is wrong, however, we will end up taking actions that we think are fitting, but are in reality detrimental.

The concept of postnormal science is meant to challenge how science is used today in attacking complex, societal questions, how society values, relates to, and uses science, and what the proper role is of scientists and practitioners of science, such as engineers, in addressing these questions. Although the postnormal concept may not be a perfect analogy or substitute for large-scale software project management, it is clear that the current reliance on a normal-science paradigm is severely wanting. At the very least, by taking a postnormal view of large-scale software projects based upon a risk-taking ethic, risk, uncertainty, and the potential impact of failure can at last be acknowledged and dealt with forthrightly, not ignored or hidden like some dark family secret.

Clearly, projects striving for goals like those of the AAS are needed, and sometimes the "performance envelope" needs to be pushed. Only through experimentation can true learning, and hence progress, take place. However, it should be remembered that one definition of insanity is when a person, failing at a task, tries the same thing over and over again, expecting a different result. Given the success rate of large-scale software projects, you decide. ●

REFERENCES

1. T. Perry, "Special Report: Air Traffic Control — Improving the World's Largest, Most Advanced System," *IEEE Spectrum*, Feb. 1992, pp. 22-36.
2. T. Kuhn, *The Structure of Scientific Revolutions*, University of Chicago Press, Chicago, 1962.
3. S. Funtowicz and J. Ravetz, "PostNormal Science: A New Science for New Times," *Scientific European*, Mar. 1992, pp. 95-97.
4. D. Parnas, "Parnas: SDI 'Red Herrings' Miss the Boat," (Letter to the Editor) *Computer*, Feb. 1987, pp. 6-7.
5. T. S. Bateman and C. P. Zeithaml, *Management: Function and Strategy*, Irwin, Boston, 1990.
6. H. Sapolsky, *The Polaris System Development*, Harvard Univ. Press, Cambridge, Mass., 1972.
7. C. Jones, *Patterns of Software Systems Failure and Success*, International Tomson Computer Press, Boston, 1996.
8. B. Boehm, "A Spiral Model of Software Development and Enhancement," *Computer*, May 1988, pp. 61-72.
9. *Encyclopedia of Software Engineering*, vol. 1, J. Marciniak, ed., John Wiley & Sons, New York, 1994.
10. R. Charette, *Applications Strategies for Risk Analysis*, McGraw-Hill, New York, 1990.
11. "Federal Aviation Administration's Advanced Automation System Investment," Government Accounting Office, GAO/T-IMTEC-88-2, March 31, 1988.
12. R. Charette, *Software Engineering Risk Analysis and Management*, McGraw-Hill, New York, 1989.
13. R. Charette, "On Becoming a Risk Entrepreneur," *American Programmer*, Mar. 1995, pp. 10-15.
14. S. Funtowicz and J. Ravetz, "Risk Management as a Postnormal Science," *Risk Analysis*, Mar. 1992, pp. 95-97.
15. S. Covey, *The 7 Habits of Highly Effective People: Restoring the Character Ethic*, Simon & Schuster, New York, 1989.

Robert N. Charette is president of the ITABHI Corporation of Fairfax, Virginia. His interests are in entrepreneurism, unified approaches to proactive risk management, systems engineering, and very large-scale systems definition and management.

Charette is the chair of the SEI risk-management program advisory board, the author of the lead guidebook on risk management for the UK Government's Centre for Information Systems, and has written numerous papers, articles, and books on the management of risk. He can be reached at PO Box 1929, Springfield, VA 22151; 7500.1726@compuserve.com.