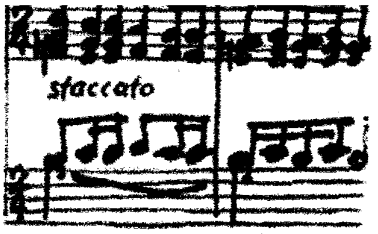


A Defined Process For Project Postmortem Review

BONNIE COLLIER, *Apple Computer*
TOM DEMARCO, *The Atlantic Systems Guild*
PETER FEAREY, *Wildfire Communications*

Most of us pay lip service to the need for software project postmortems, but the literature offers little guidance on how to conduct them. The authors propose a tentative, standard process for conducting postmortem reviews and describe activities, roles, and artifacts of the process.

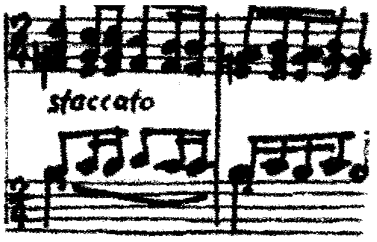
Conventional wisdom in the software industry decrees that it's good practice to conduct a postmortem study at the end of each project. The literature is liberally salted with exhortations not to miss this important opportunity to learn from our mistakes.[1-6] Some would even suggest that this is not just a useful undertaking, but one of the fundamental principles of successful software development. Alan Davis for instance lists it as

> *"Principle 172: Conduct a Postmortem . . . . At the end of every project, give all the key project players a three- or four-day assignment to analyze every problem that occurred during the project . . ."*[7]

Don Norman goes so far as to suggest that postmortem information be made part of a national repository similar to the Aviation Safety Reporting System run by NASA-Ames. The information culled from a postmortem, he argues, is so valuable that it should be shared.[8]

The rationale authors most often cite for postmortem analysis is that only by analyzing our shortcomings can we learn to do better. This idea is aptly and elegantly stated by Henry Petroski:

0740-7459/96/$05.00 © 1996 IEEE



"Signal successes in engineering have tended to arise not out of a steady and incremental accumulation of successful experience, but rather in reaction to the failures of the past."[9]

Failure, of course, is a subject on which the postmortem must dwell. As Norman observes, "the behavior we call human error is just as predictable as system noise, perhaps more so."[8] Just as a circuit designer tries to contain the effects of expected noise, we can try to approximate a statistical quality control over the human failures that plague projects: underestimation, function inflation, requirements mismatch, and so on. We must begin by cataloging such failures and learning from their patterns. The postmortem plays a key role in this process.

The postmortem takes aim at those areas with maximum potential for improvement.

WHY A DEFINED PROCESS?

The focus on human error explains the attraction of conducting a postmortem; it also explains why we so often neglect this key activity and thus fail to learn from past mistakes. For example, in a survey of 92 medium-sized MIS organizations, Kuldeep Kumar found that more than one fifth did no postmortems whatsoever. Of the companies that did conduct them, more than half did so on fewer than half of their projects.[10] When asked why more postmortems were not conducted, managers trotted out the usual suspects: unavailable staff, shortage of qualified analysis personnel, no agreed-upon criteria for evaluation, pressures of work, and so on. To complicate matters further, the organizations that elected not to participate in Kumar's study (79 percent of those asked) were almost certainly less prone to do postmortem analysis. Even the least jaded system developer harbors a grim suspicion that the projects not subjected to postmortem analysis are the very ones we could learn the most from.

Concern about frank analysis of failure creates a natural disincentive within the organization to conduct a postmortem; it also creates apprehension in the individual preparing to take part in ones that are held. Both of these effects can be alleviated by a well-understood, defined process that is put into place before the event.

A few organizations have shown signs of having such defined processes,[11] but have an understandable reluctance to share the details: Any company that might someday end up in court is wary about calling attention to the way it captures information about its failures. The result is that postmortem procedures are continually being invented anew as each organization tries to define a way to conduct post-project reviews. The postmortem exists to help us learn from past successes and failures, but, ironically, we haven't yet come up with a way to learn from past successes and failures in developing the postmortem processes.

WHY THIS PROCESS?

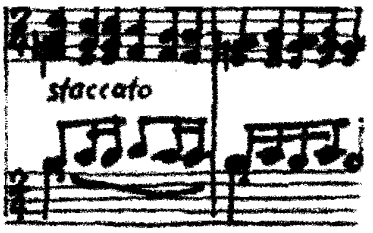
The goal of a postmortem is not so much to gain insight into events that impede software development as to use the analysis results to improve methods and practices. Discovering which behaviors need changing is not a trivial task in complex systems, particularly on large, lengthy projects. In our proposed process, we use proactive problem-solving tools[12] that are well-suited to examining complex systems and guiding project teams to identify the few vital improvements that will help them attain their goals. But by itself, conducting a postmortem is no guarantee that beneficial change will occur: We have seen projects put out volumes of postmortem findings (80 pages or more) with results so unstructured and vague as not to be actionable. The method in this madness definitely matters.

We decided, therefore, that our defined process had to:

- Put in place a set of documented, well-understood procedures and guidelines that would be available to all participants prior to the postmortem.
- Establish communication channels that would elicit even difficult findings without compromising individual safety.
- Make clear to all participants that the process would be positive and blame-free.
- Respond to the common concern that postmortem results are destined for a write-only repository and have no effect on future projects.
- Provide an appropriate balance between the cost of postmortems (precious people time) and the return on investment. Part of the return on investment would include therapeutic benefits, as well as insight into root causes and their effects and actions taken — real changes in behavior on the part of the organization.

THE WHOLE AND ITS PARTS

We have each worked with a handful of carefully selected defined processes for conducting postmortem and "lessons learned" sessions and applied them in a dozen or so organizations. We have involved more than 1,300 project members on some 22 projects in postmortem events. This experience gave us a rare opportunity to write down a defined process that would be useful to others. We combined elements of our individ-



ual experiences to come up with an aggregate, five-part defined process:

- *Design and promulgate a project survey*, a mechanism for collecting project information without compromising the confidentiality of our respondents (potentially all project participants).
- *Collect objective project information*, data that reveals the project's health, such as resource costs, boundary conditions, schedule predictability, defect counts, and so on (although such information is largely objective, we use some subjective scaling to capture it).
- *Conduct a debriefing meeting*, a structured gathering of project members aimed at collecting information that the survey has missed (and organized in part to allow people to vent).
- *Conduct a Project History Day*, a meeting of a selected subset of project participants aimed at reviewing project events and all project data, leveraging all input from the previous steps, and using Deming-like quality management methods[12] to discover key insights into the project dynamics and its driving forces.
- *Publish the results*, a report to the organization focused on using postmortem lessons learned to guide organizational improvement.

Finally, the postmortem must have a well-understood link to the conduct of future projects. This link must be in the form of a commitment by management.

We now describe the five steps and suggest a form for this management commitment. Although you may elect to invent your own defined process for postmortems — one that is tailored to your project's size and the particular needs of your company — we hope that our process can be a starting point for your invention, a way to avoid beginning with a blank sheet of paper.

STEP 1: PROJECT SURVEY

Surveys, particularly if they are implemented electronically, are a relatively quick and painless way to get the project team's input about a wide variety of project-related topics. Survey results can inform and guide the rest of the process in two ways: It can help focus the subsequent meetings by establishing scope and severity of the major issues and it can provide the organization with some quantitative, cross-project data that can be used to track improvement over time. For example, if a given project reports that communications about status were ineffective, we can use that information to help project management determine areas to improve, and then watch this piece of data on successive project postmortems to see if the changes we made brought improvement.

For large projects, electronic surveys allow team leaders to collect volumes of timely feedback. The postmortem team in one of our organizations used a HyperCard tool to generate an electronic survey, which was then posted on a server. Team members downloaded the file to their local system and answered the questions, including comments and demographic information. When the stacks were closed, the results were automatically returned to the originator without identifying the respondent.

Responding to such a survey takes 30 minutes to an hour, depending on the number of questions, response time, and the respondent's need to comment. Projects using this scheme have experienced survey return rates as high as 20 to 30 percent. We suspect that many people who would not take the time to fill out a paper survey or come to a debriefing meeting will respond to the electronic survey. Beyond ease of use, the most likely driver behind this high return rate is the anonymity that the surveys provide. Once team members understand that the process allows (even solicits) negative feedback along with the positive, and that the source of the feedback cannot be identified, their sense of security and participation increases.

Survey design. The first requirement of designing a project survey is to determine what questions to ask. Crafting survey questions that are simple, direct, and precise is a challenging task. In addition to skills in attitude-scale construction,[13] you must have a clear understanding of the major issues and dynamics that determine project success or failure. Some categories of questions are common to most large software projects; a generic set of such categories is presented in the box on page 68, along with sample questions.

Most projects start out with a list of questions that is far too long. To filter the list, we subject questions to tests like these, which are designed to call attention to empty or useless questions:

- What does this question buy us? What will we do with the information gained from the responses?
- Do we already know the answer?
- Does the question appear to point a finger at a specific group or person?
- Does the question appear to be leading?

POSTMORTEM RESOURCES ON THE INTERNET

Certain copyright-free documents and samples related to our defined postmortem process are available at http://www.wildfire.com/research/postmortems.html. Among the items on the site are

- a concise, defined process in a form suitable for adoption or adaptation;
- a sample survey;
- a sample tabulation of survey results;
- tips and tricks for facilitating the various process steps;
- sample affinity diagrams; and
- a schedule-predictability tool.

SAMPLE SURVEY QUESTIONS

Our sample survey that is posted on the web contains a dozen or so questions in each of the eight categories, such as

Category One: Support and Goals

Sample question: Were interdivisional lines of responsibility clearly defined throughout the project?

[ ] Always [ ] Sometimes [ ] Rarely [ ] Never

Category Two: Expectations and Communications

Sample question: Did project-related meetings make effective use of your time?

[ ] Always [ ] Sometimes [ ] Rarely [ ] Never

Category Three: Issues Resolution:

Sample question: Were you empowered to participate in discussions regarding issues that affected your work?

[ ] Always [ ] Sometimes [ ] Rarely [ ] Never

Category Four: Information Access

Sample question: Did schedule changes and related decisions involved the right people?

[ ] Always [ ] Sometimes [ ] Rarely [ ] Never

Category Five: Product Specifications

Sample question: Was project definition done by the appropriate individuals?

[ ] Always [ ] Sometimes [ ] Rarely [ ] Never

Category Six: Engineering Practices

Sample question: Was the build process for the component area you worked on effective?

[ ] Always [ ] Sometimes [ ] Rarely [ ] Never

Category Seven: The Big Picture

Sample question: Considering the time-to-market constraint, were the right tradeoffs made between features, quality, resources, and schedule for this product?

[ ] Always [ ] Sometimes [ ] Rarely [ ] Never

Category Eight: Demographics

Sample question: What was your primary function on this project?

[ ] Quality Assurance [ ] Development [ ] Marketing

[ ] Project Managment [ ] Documentation

♦ Can we get this information from another, currently documented source?

After the list has been reduced to a manageable set, it is time to scrub each question: remove ambiguities, check for common terminology, and examine the language for biased statements, style consistency, and so on. The survey is now ready to be sent.

Evaluating results. As the results come back, the work turns to tabulation and evaluation. We usually look at the number of returns for each question and the percent of responses for each answer. When the same questions are used for multiple projects, organizations can begin to use the data to build and test theories about their software-development processes.

So far, the input has been based on the opinions and perspectives of team members. In the next step we complement these subjective findings with objective data.

STEP 2: COLLECT OBJECTIVE INFORMATION

Critical to any improvement effort, metrics let a team know whether it has met or exceeded its improvement and project goals. Metrics captured during the project are collected in the second step of the postmortem process for four reasons.

♦ *The postmortem must focus on those areas that will provide the biggest improvement opportunity.* Data collection helps provide the direction the team needs to ensure a high return on its time investment. Collecting simple data will provide teams both an indication of what the real problems were and also the magnitude of the problems. This knowledge will focus the remaining investigation efforts.

♦ *Data collection across projects allows comparisons across multiple projects.* This will eventually let us see the effect of improvement efforts. When a small change is made in the way defects are isolated, for example, we can note if the change had an effect on closure time and, if so, the magnitude of the effect.

♦ *Project data provides information for subsequent scheduling efforts.* Though schedule setting is and will remain an imprecise science, individuals do get better at it over time. Common pools of coherently collected data facilitate this learning process.

♦ *Hard data helps focus discussion.* An issue is less debatable when it is grounded in actual information rather than opinions and assumptions.

Metric types. Three kinds of metric are important in the postmortem process: metrics of cost, schedule, and quality. Each of these represents a distinct element of the project. Here are a few sample metrics, by category.

Cost metrics:

♦ person-months of effort for each of the major roles: development, quality, project management, marketing, documentation, and so on;

♦ total lines of code by function;

♦ number of lines of code changed or added by function; and

♦ count of interfaces, and interfaces added, changed, or deleted.

Schedule metrics:

♦ original schedule,

♦ history of schedule-slippage events, and

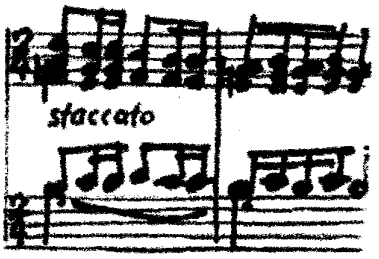
♦ analysis of schedule predictability.

Quality metrics:

♦ defects at each stage of development, and

♦ defect find and closure rates.

Tracking. There are many methods for tracking these items. When the data has a subjective element, we provide a scale to choose from along with some examples for the high and low values. Given the subjectivity of the measures, we found it valuable to collect data three times during the project: at the beginning, middle, and end. This helped us understand what was



over- or underestimated. The meaningful distinction with objective data is whether or not there is a tool to aid in the collection. We have found no successful way to collect data without the aid of a tool.

The data collected during this second step serves as input to the remaining steps.

STEP 3: DEBRIEFING MEETING

Debriefing meetings provide team members with the opportunity to give some direct feedback about the project — including an in-depth view of what did and did not go well — in a structured, safe environment. It is an opportunity for project leadership to probe deeper into observed effects — positive and negative — and begin to track down their root causes. Positive results like improved processes or policies are also captured. Survey results guide the topics covered in the meeting, but participants often bring up new issues to cover. The entire project team should have an opportunity to participate. However, there should be no more than 20 to 30 people in any single meeting. Larger projects may require a series of meetings.

Key roles. We have found that debriefing meetings succeed best when three roles are filled:

- *Chair:* encourages attendance, illuminates issues or questions that arise and provides technical support for the facilitator. The chair should be knowledgeable about the project and its issues. The chair is usually a key player on the project — a member of project leadership or functional management who represents the content/functional areas being debriefed.
- *Coordinator:* provides the infrastructure needed to support the meeting. The coordinator schedules the meeting, recruits the facilitator, books the conference room, and ensures that the appropriate supplies are on hand. During the meeting, the coordinator acts as scribe.
- *Facilitator:* leads the meeting, providing focus, direction, and clarification, and ensures that participants feel safe. Preferably, the facilitator is not part of the project team being debriefed.

Risks. Although potentially useful, the debriefing meetings entail certain risks that need to be recognized up front:

- Unless guided by a skilled facilitator, debriefings can evolve into "dumping" sessions by disgruntled team members who monopolize precious time and often fixate on a single issue at the expense of other, more critical ones.
- Managers and project leadership can, by their language and defensiveness, establish an atmosphere that inhibits participation.
- Failure to capture and use debriefing information can also be a problem. Participation in debriefing meetings may be limited, but most projects — even healthy, successful ones — can have debriefings that produce a plethora of output. A project team of 150 members who hold a dozen debriefing meetings can produce hundreds of flip-chart pages of issues, comments, and solutions. Failure to capture is often the result of the facilitator attempting to serve as both scribe and facilitator, and thus being unable to capture the volumes of output.

Key benefits. Despite the potential pitfalls, debriefing meetings encourage wide participation and provide perhaps the most important postmortem by-product: By allowing team members to vent, debriefing meetings can provide a cathartic, even therapeutic effect. These pseudo-ceremonial meetings can cleanse the air, empty old baggage, and give team members the hope and courage needed to attack the next project.

Beyond information. At this point in the process, leadership teams are nearly submerged in information about the project. With survey results, objective data, and debriefing minutes in hand, they now have the means to achieve substantial insight into problems and symptoms — but may be overwhelmed by the magnitude of information. The next step of the postmortem process provides a mechanism to make sense of the data mass and build a common vision of the real causes of observed problems.

STEP 4: PROJECT HISTORY DAY

The Project History Day is perhaps most important step of the postmortem process. (See Alan Graham's article "Organizational Learning: Medical Metaphor and Corporate Practice" for a description of the project history method.[14]) By combining reflective analysis of project events with a rich collection of actual project data, history-day participants now move the focus from the general to the specific — and eventually to a set of root causes that can be acted upon.

Problem statement. Guided by information collected in the previous steps, project leaders now formulate a problem statement that becomes the focus of the day's activities. For example, a project that was plagued with significant schedule slippages might use this problem statement: "What are the root causes of events that determined or affected resources, schedule, and quality?" Posted on the wall throughout the day,

Collect feedback from all levels of your organization.

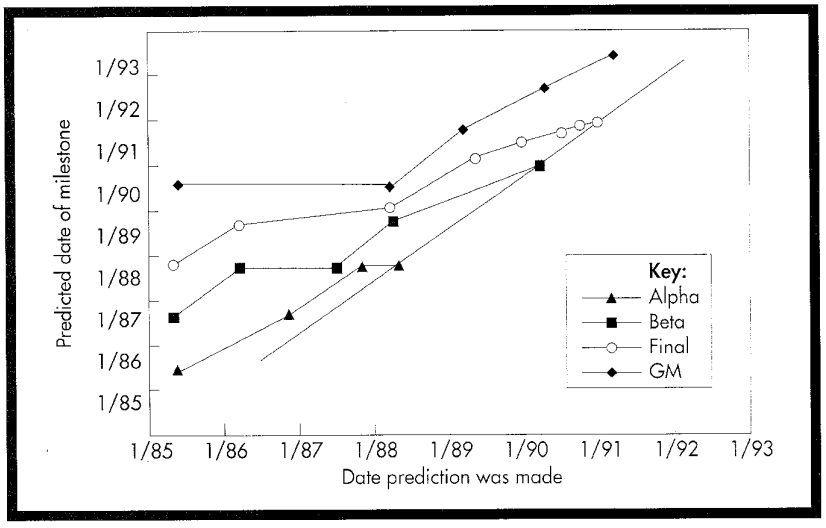


Figure 1. *This sample schedule-predictability chart plots the predicted milestone date and the date the prediction was made. The ideal is a horizontal line near the horizontal axis, indicating short time to market and a schedule date that never slipped. The chart can be used to estimate the actual start of production based on slip rate. It also shows the amount of optimism about schedules at the beginning of projects.*

this question becomes a constant reminder of the investigation's focus.

Key team players who understand the issues relevant to the stated problem and their root causes are now brought together in the Project History Day meeting. A facilitator, experienced in root-cause analysis and quality-management[12] tools, takes charge of the day. Ideally, the event should be scheduled for four to six hours. It may take more or less time depending on the diversity of the groups' perceptions, the project length, and the number of issues.

Participants. Unlike the prior steps, this activity is not for the entire project team. Assembling the right participants is critical to the day's success. In the example above, for instance, the team should include those who not only know the events that affected schedule, quality, and resources, but also know why the events happened. The participants and observers should have a deep understanding of the issues and decisions that drove the project, as well as the reasons behind those decisions. This could include members of the engineering group (quality and development), the marketing team, the project management team, and any others involved in the project's most challenging problems. Ideally, the group should be limited to six or eight people.

Activities. As the Project History Day begins, participants should arrive with project events fresh in their minds: They have been given the complete set of postmortem information gathered to date and have reviewed their private stores of status reports, e-mail, meeting minutes, and so forth. The first step is to examine a schedule-predictability chart — a line graph depicting all predicted and actual milestones. Figure 1 shows an example of this chart.[15] After studying the chart, the team works together to develop a detailed timeline of significant project events from the initial concept (when did the project really start?) to product introduction. As they examine the timeline, they are searching for those key events that meet the criteria of the search identified in the problem statement. As those events are identified, the team completes root-cause analysis on each event, asking in each case, "Why did this occur?" and looking for the causes of events and the causes of the causes. This question is repeated until there are no new answers. Before long, there are at 60 to 100 candidate root causes posted on the event timeline.

As bits and pieces of the project problems are plucked from memory and posted around the room, the team begins to form a new and deeper insight, and, increasingly, a joint insight into their product-development system. At this point, teams may finally be able to see the "bleeding elephant" smack dab in their midst; that nagging problem that no one was able to name before — much less solve — becomes gradually evident to all. Project History Day, when it works well, is like finally being able to see the image hidden in a random dot stereogram or Magic Eye picture. Like the stereogram viewer who suddenly catches on to the trick that lets the image appear, history-day participants now experience a gratifying "Aha!"

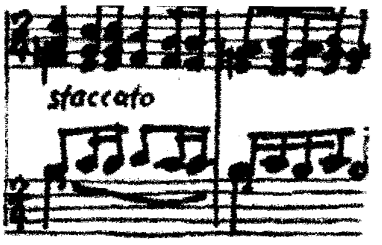
Results. To capture results, the team identifies and describes on index cards or "sticky notes" the top 20 or so root causes — the best answers to the problem statement. These casual statements are scrubbed for clarity and specificity and grouped intuitively using an affinity diagram tool.[12] The groupings are labeled, then prioritized by the team. The last task of the day is to organize the groupings by causal relationships.

Project History Days are psychologically and mentally exhausting, but extremely fruitful. The results are usually clearly actionable and there is strong consensus among participants that the conclusions ring true. Now that the team knows what happened on the project, it is time to share the lessons learned with the rest of the organization.

STEP 5: PUBLISH THE RESULTS

By the end of the Project History Day, the team has developed substantial insight into the project's software-development process. But insight alone doesn't buy much in the marketplace; the postmortem can only be judged a true success if that insight is turned into action. The last step of the postmortem process is focused on recycling lessons learned into improvements in the development process.

In this step, the leadership team summarizes its findings and publishes the summary in the form of an "Open Letter to Project Teams." The audience



for the letter is management, project participants, peers, and the other project teams in the organization. The open letter is made up of four parts:

♦ *Project Description:* a brief project overview including the type of product and pertinent information related to the unique circumstances of the development effort.

♦ *The Good:* a summary delineating the positive findings identified in the postmortem. Usually, these are infrastructure improvements, process changes, and tools that were developed during the project.

♦ *The Bad:* a summary of the three worst factors that impeded the teams' ability to meet goals. These are often the top three items listed in the last step of the Project History Day.

♦ *The Ugly:* a prescription for improvement. Typically, the postmortem team selects one key issue or problem — the one thing that is so important that it must be fixed before they start work on another project. They provide a clear and precise problem description so that everyone will be able to observe if and when it is truly fixed. In the improvement recommendation, the team includes specific metrics to capture the extent of the problem and reveal changes as things improve.

The postmortem is now complete, but the organization's due diligence does not stop here. Long-term success requires that the organization now go that extra mile to turn what "everybody knows" into what "everybody does." Analysis of postmortem results often indicates that changes in practices must occur at all organizational levels, from executive management to individual contributors.

To insure that the organization profits from postmortem lessons learned, we've compiled a few suggestions.

♦ Store all postmortem output — including survey results, debriefing minutes, objective project data, schedule-predictability graphs, affinity documents, and the postmortem summary — in a central repository accessible to everyone in the organization.

♦ Categorize all lessons learned by functional area or the process they affect. Assign each item to a specific person on the next project, and charter that person to investigate the lesson learned and report back to the leadership team on whether or not it is a risk for the new project, and if so, how the risk will be addressed.

♦ Present the results of one postmortem at each of upper management's regularly scheduled organizational reviews. Have leadership team members present the open letter from their postmortem and the results of the affinity-diagramming activity in their own words. There can be a powerful magic in these postmortem stories: Managers can't help but gain insight into how they contribute to problems that impede project development.

♦ Assign each lesson learned to a person in the organization who will be held responsible for investigating and implementing a solution. Ultimately, no changes will occur in the organization unless someone is held responsible.

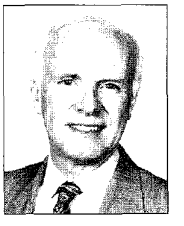
The success of the postmortem— or of any learning process— demands a context that makes organizational learning possible.[6] Management must make an honest and sincere commitment to establish this context. This commitment should take the form of a public resolution to implement risk management on subsequent projects and to make all postmortem findings input to that risk-management effort. After all, lessons learned the hard way on past projects are, if nothing else, risks for future projects. Participants are empowered when they know that each issue raised during the postmortem process must be added to the risk database and evaluated methodically on each subsequent project. ◆

REFERENCES:

1. T.K. Abel-Hamid and S.E. Madnick, "The Elusive Silver Lining: How We Fail to Learn from Software Development Failures," *Sloan Management Review*, Fall 1990, pp. 39-48.
2. V. Basili and H.D. Rombach, "The Experience Factory," *Encyclopedia of Software Eng.*, John Wiley & Sons, New York, 1994, pp. 469-476.
3. J. Boddie, "The Project Postmortem," *Computerworld*, Dec. 7, 1989.
4. E. Chikofsky, "Changing Your Endgame Strategy," *IEEE Software*, Nov. 1990, pp. 87, 112.
5. M. Page-Jones, *Practical Project Management*, Dorset House, New York, 1985.
6. K. Clark and S. Wheelwright, *Managing New Product and Process Development: Text and Cases*, The Free Press, New York, 1993.
7. A. Davis, *201 Principles of Software Development*, McGraw-Hill, New York, 1995.
8. D.A. Norman, "Commentary: Human Error and the Design of Computer Systems," *Comm. ACM*, Jan. 1990, pp. 4-7.
9. H. Petroski, "History and Failure," *American Scientist*, Nov.-Dec. 1992, pp. 523-526.
10. K. Kumar, "Post Implementation Evaluation of Computer-Based Information Systems: Current Practices," *Comm. ACM*, Feb. 1990, pp. 203-212.
11. S. Brady and T. DeMarco, "Management Aided Software Engineering," *IEEE Software*, Nov. 1994, pp. 25-32.
12. S. Shiba, A. Graham, and D. Walden, *A New American TQM*, Productivity Press, Portland, Ore., 1993.
13. J. Sperry, *How to Measure and Improve Customer Satisfaction: Attitude Scales*, Quality Alert Inst., New York, 1993, pp. 10-16.
14. A.K. Graham, "Organizational Learning: Medical Metaphor and Corporate Practice," *Learning Organizations*, Productivity Press, Portland, Ore., 1994, pp. 452-460.
15. J. Carter, *Project Histories*, Product Development Consulting, Boston, Mass., 1992.

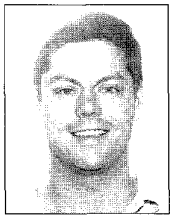
Bonnie Collier is a manager in the Apple New Product Process program office at Apple Computer, where she discovers, defines, leverages, and supports best practices in the product-development process. In addition to using best practices in software-engineering from the industry and from within Apple, she relies heavily on TQM principles and tools for defining improvements to product development.

Collier received a BA in speech, hearing, and language pathology and audiology from the University of Houston, with postgraduate work and certification in special education and education technology. She was certified in TQM at San Jose State University in 1994.



Tom DeMarco is a principal of The Atlantic Systems Guild, a computer-systems think-tank. He has written several books, including *Why Does Software Cost So Much?* and *Peopleware: Productive Projects and Teams* (with Tim Lister). DeMarco served on the ESS-1 project at AT&T Bell Labs, managed real-time projects for La Cegos Informatique, and participated in the installation of distributed, online banking systems in Sweden, Holland, France, and Finland.

DeMarco received a BSEE from Cornell University, an MS from Columbia University, and a diplome from the University of Paris at the Sorbonne.



Peter Fearey is a project manager at Wildfire Communications. His interests include improving Wildfire's software-development process and building management tools to help analyze project status and team performance. He was previously employed at Apple Computer, where he worked on System 7.1 and System 7.5.2 and was a cofounder of the System Software Process Team.

Fearey received a BS in computer science and economics from Dartmouth College.

Readers can address questions about this article to Collier at Apple Computer, Inc., 1 Infinite Loop, MS 306-1JS, Cupertino, CA 95014-2083; Collier.Bo@Applelink.Apple.Com.