Pro-Active Metrics

George H. Wedberg
McDonald Bradley, Inc.

Metrics can be used for more than measuring things. They can be viewed as the starting point for process improvement on a project and as a mechanism to improve communication and teamwork. This article summarizes what was learned in this regard over the course of several years as the metrics program was developed and managed for the U.S. Army Standard Installation/Division Personnel System – Version 3 (SIDPERS-3) project.

Project management metrics typically are used to measure the progress of a project and the quality of its output. They also may be used to monitor key parameters of the development process—for example, the stability of the requirements or the effectiveness of the technical review process. The principal benefit of a metrics program is improved *control* of the project; metrics furnish an overview of progress against plan, provide early warning of problem situations, and enable management to take corrective action.

That control can be significantly enhanced by taking a more pro-active approach to the metrics program—an approach in which the information obtained while gathering metrics data is used to improve the processes used by the project and the effectiveness of the project team. The metrics team is in a good position to observe the lower-level workings of the project, including the problems and inefficiencies that are hindering the progress of the developers and whether they are being addressed effectively. Information of this kind is an invaluable source of ideas for process improvement. It is also a starting point for building the management-developer bonds needed for a successful project.

The pro-active approach to metrics also focuses on communication. The success of a project depends, to a significant degree, on how the participants feel about the project, each other, and their management. People need to feel that they are accomplishing something of significance, that they can rely on each other and their management, that they are kept informed about project issues, and that their concerns are heard. Much of this can be accomplished by relatively simple communication mechanisms—particularly if that communication is backed up by action.

A successful project usually is the result of marrying technology and psychology. Both are necessary; neither is sufficient by itself. Technology usually prevails; we devote enormous energy to selecting the optimum software tools and hardware platforms. But we cannot forget that people want to work in a positive environment where they can make a solid contribution, exercise their creativity, and develop their skills.

During the course of the SIDPERS-3 project, we gradually evolved a pro-active approach that involved five specific actions designed to help create that type of positive environment on a team that peaked at 145 people. These actions were designed to involve the developers in the metrics process, to ensure they were heard with regard to process improvements, to improve communication at all levels, and to strengthen teamwork on the project.

Metrics Criteria

We selected the metrics to be used on the project and the way in which they would be used with an eye toward the climate we were trying to create. We understood that no one likes to be measured and that metrics are threatening enough as it is, so the intent was to not make it any worse. People who feel threatened pull into their shells and become defensive; they tend to tell you only what you want to hear. We wanted a climate in which they came out of their shells and told us everything.

We ensured that the metrics used also provided information that was useful to the development teams and that it was provided in a timely enough fashion that they could track their own progress. The metrics process was set up so that most of the data came from the developers; this helped ensure that the biweekly metrics reviews focused on progress and problems rather than on disputing the data. And finally, we made the metrics as nonpunitive as possible. The management team did not use them to judge people; metrics were not used as the basis for blame, threats, or performance reviews. Metrics were not used to compare teams; some of this is inevitable, but management did not encourage it.

The metrics team (consisting of one full-time person and two others part time) served as the proponent for this approach to metrics. We articulated the overall intent, convinced management to adopt it, and successfully argued the case, for example, with those who initially saw metrics as a convenient way to measure people rather than project accomplishments.

Biweekly Metrics Review

Every two weeks, the senior management met with the eight to 10 development team leaders and the metrics team to review the project metrics. Over time, we moved these meetings away from the initial inquisition-excuses-blame mode to a more positive mode in which we focused largely on removing the impediments that kept one team or another from meeting its planned objectives.

We changed the meetings by both convincing management of the advantages of a solution-based approach and taking a pro-active approach to identifying and removing impediments to progress. The problems faced by the teams were real: hardware that arrived late, repeated compiler problems, interteam process problems that necessitated excessive rework, and many more. Before each meeting, we asked

the team leaders to identify their current problems in writing. We addressed each problem in the meeting, developed a plan for dealing with it, and followed up after the meeting to ensure the plans were being carried out. The senior managers agreed to help effect timely resolution of the problems and left many review meetings with a list of calls to make.

In time, the development team leaders realized that putting a problem on the table would result in help rather than blame, and this encouraged them to surface problems rather than hide them. Giving the project manager information about real and potential problems strengthened his control of the project in at least two ways. First, he could deal with real problems while they were still small and relatively easy to resolve. Second, he could investigate potential problems and take mitigating action long before they affected the project. His control also was strengthened because team leaders and senior management were now working as a team with a corresponding increase in mutual trust. That the latter was not easily quantified made it no less real.

Implicit in this teamwork was the unspoken understanding that once the impediments to your progress have been eliminated, there is no longer any reason you should not make your dates.

Technical Reviews

Technical reviews in one form or another, e.g., walk-throughs and inspections, have long been acknowledged as technically useful. They are used to find problems early and to help prevent future problems. Don O'Neill described a software inspection program and its benefits in "Setting Up a Software Inspection Program." [1]

As one response to the number of software defects showing up in the metrics, the metrics team encouraged the development of a formal code inspection process by the software developers. The resulting process used extensive checklists, assigned roles and responsibilities for the review, and tracked the numbers and types of defects found.

The code inspections also increased the sense of teamwork on the project. Almost any well-run technical review process accomplishes this for several specific reasons. First, the process teaches people how to depend on one another, which is a key element of teamwork. They learn that accepting detailed evaluation of their material by their peers leads to a better product. They experience the satisfaction of being heroes when it counts most—at product delivery time.

The barrier that has to be overcome is ego. Few enjoy having their mistakes and bad assumptions pointed out, particularly in a group meeting. A good technical review process minimizes this problem by decriminalizing errors [2]. Errors are treated as a fact of project life; everyone makes them, no one is blamed for them. The idea is to make them visible, classify them, and learn from them. In time, people learn to react to their errors objectively rather than defensively, and a great battle has been won. The importance of subjugating ego cannot be overemphasized. A 1996 article in *Fortune* magazine described a number of world-class teams—from the U.S. Navy Seals to the Tokyo String Quartet. A common theme throughout the article is the lack of individual egos on these teams; every member is totally focused on the mission of the team [3].

The technical review process also builds teamwork because participants learn a new and useful process together. They learn how to evaluate a product against written requirements rather than personal preference. Experience with inadequate and incomplete requirements also leads to enlightenment about developing good requirements. Participants learn how to classify errors according to their source, which is a basis for preventing future errors. Participants also learn how to run a disciplined meeting, e.g., the purpose of the review is to identify errors, not resolve them; resolution of errors is the privilege and responsibility of the producer of the material [4]. In time, the participants learn that they can be more successful together than as individuals.

Empty rhetoric about "teamwork" is widespread in the business community; technical reviews are one concrete way to implement the rhetoric.

Communication

The goal of our communication processes was to establish as open an environment as possible—one in which people were comfortable surfacing problems and telling us what they needed to be more effective. We used several mechanisms in addition to a lot of one-on-one discussion:

- We conducted an informal written survey of all members of the project that asked them what they thought was going well and not so well, what problems they were having, what they would like to see changed, and the like. The objective was not measurement but rather to see what they had to say. Although the survey was anonymous, we invited those who were concerned about having their E-mail traced back to them to respond any way they chose, including notes under my door at night. Because a few did the latter indicates the difficulty of establishing trusting communication.
- We conducted skip-level meetings in which programmers, testers, etc., met in groups of a dozen or so with the project manager—no other managers or team leaders were present. The participants were encouraged to bring questions from their teammates as well. This was an opportunity to discuss rumors about high-level topics such as project direction and funding, to surface frustrations directly to senior management, and to understand and influence the project manager's thinking. This direct contact, somewhat unusual on a project of this size, let the project manager explain his priorities and reasoning without the usual middle management filters and gain a direct understanding of what was important to the employees he supervised. Out of these meetings also came a number of "social" changes to the project, e.g., the occasional holiday party evolved

into monthly project luncheons, and casual Friday became casual summer, then became permanent casual.
- We conducted lessons-learned exercises after each major increment of the project. The metrics group collected inputs from all project areas, organized them by subject area, and facilitated working meetings to discuss problems, working relationships, and potential process improvements. The documented outputs of the meetings were the basis for process improvement activities.

The communication mechanisms, in addition to revealing problems, also raised the level of trust on the project. As might be expected, the developers sometimes used their communication opportunities to challenge management, in effect, to improve some aspect of the project. When management responded promptly with meaningful actions, both sides knew that everyone's level of commitment had just risen.

Process Improvements

The biweekly metrics reviews, the lessons-learned exercises, and the other communication mechanisms brought out problems and frustrations of all kinds—from inadequate technical planning to lack of vendor support to poor working relationships between project teams. All such concerns were analyzed to determine their underlying causes. More often than not, the cause was a poor or nonexistent work process; this was particularly true of problems involving friction between project teams.

Of course, the configuration management (CM) team was unhappy that developers were turning over code to them before it was properly integrated; investigation showed there was no well-defined process for integration and turn-over. And the coders were annoyed at having to rewrite modules numerous times because the database kept changing; investigation showed there was no visible plan or schedule for database changes.

Some of the resulting process improvements were accomplished simply, e.g., a verbal agreement between two individuals to exchange key information at regular intervals. Other improvements required analysis, documentation, and review. The CM problem noted above led to a diagram showing the sequence of all steps required to manage software modules from the developer's unit test onward; this included the definition of each step, the person responsible for accomplishing it, the machine on which it was to be performed, and so forth. Project-wide distribution of that diagram significantly reduced the friction between the developers and CM while increasing project productivity. This, like most process improvements, strengthened the project manager's control of the project by making the software development process more predictable.

Conclusion

The pro-active approach to project management metrics—a combination of measurement, process improvement, and communication—strengthened the project manager's control of the project in several ways.

- The measurements provided an objective picture of the project's progress, status, problems, successes, and failures. This factual information provided the basis for subsequent management and technical decisions.
- Process improvements made many aspects of the project more predictable. Every process improvement that standardized a procedure or eliminated an impediment, for example, made it more likely we would get the desired result in the expected time frame.
- The open communication environment meant that we got useful input from a wide variety of people at all levels of the project. We also believe, but cannot prove objectively, that there were many intangible benefits from that environment—people who are both informed and heard, and thereby involved, have a higher level of commitment. We saw that commitment on many occasions.

Each of these practices emphasizes the role and needs of the project's individual contributors and thereby strengthens their connection to the project and its success. It is difficult to remain disinterested or cynical when the work is going well and the project is helping you meet your personal goals. ◆

About the Author

George H. Wedberg is a program director with McDonald Bradley, Inc. He has over 20 years experience with all aspects of the software development lifecycle, in both the federal and the commercial sectors. Past accomplishments include development of the first software engineering program for General Electric Information Services Company, development of the metrics and risk management programs for the SIDPERS-3 project, and creation of measures of effectiveness for the Department of Health and Human Services and for the U.S. Marine Corps. He holds a doctorate in physics from Indiana University.

McDonald Bradley, Inc.
Suite 805
8200 Greensboro Drive
McLean, VA 22102
Voice: 703-827-9376
Fax: 703-827-8604
E-mail: wedbergg@erols.com

References

1. O'Neill, Don, "Setting Up a Software Inspection Program," *CROSSTALK*, Software Technology Support Center, Hill Air Force Base, Utah, February 1997.
2. Rifkin, Stan and Charles Cox, "Measurement in Practice," *Carnegie Mellon University Software Engineering Institute Technical Report CMU/SEI-91-TR-16, ESD-TR-91-16*, July 1991.
3. *Fortune*, Feb. 19, 1996, pp. 90-99.
4. Freedman, D.P. and G.M. Weinberg, *Handbook of Walkthroughs, Inspections, and Technical Reviews*, Dorset House, New York, 1990.