Implementing Effective Software Metrics Programs

TRACY HALL, University of Westminster
NORMAN FENTON, City University

Increasingly, organizations are foregoing an ad hoc approach to metrics in favor of complete metrics programs. The authors identify consensus requirements for metric program success and examine how programs in two organizations measured up.

Until relatively recently, software measurement in the Western world has been a rather *ad hoc* affair focused on measuring individual, product-based software attributes. This rather one-dimensional approach to software measurement is now changing. Increasingly, organizations are integrating complete software metrics programs into their software development processes. That is, they are habitually using a balanced range of product and process measures in their micro and macro decision making systems.

There are several reasons that partially explain the recent move toward complete metrics programs. Not only has the process improvement bandwagon raised a general awareness of metrics, but to reach higher Capability Maturity Model levels, organizations must incorporate metrics into their development process. Moreover, there now appears to be consensus about the need for

0740-7459/97/$10.00 © 1997 IEEE

TABLE 1
ORGANIZATIONAL DETAIL

Characteristic	Embedded Systems	Information Systems
General profile	Engineering company	Large public-sector organization (in the process of transferring into the private sector)
Applications	Defense-related Embedded control software 90% of applications are safety-critical	Online data processing systems
Development environment	Variety of advanced approaches, including formal methods and code, analyzing tools	State-of-art in using new methods Keen on using project management methods and tools
Quality framework	Strong (on the surface), but suboptimal use of reviews and inspections AQAP certified (defense quality certificate)	Well-used framework Consistent use of basic quality controls In the process of seeking software quality certification
Management framework	Very complex staff structure Steep hierarchy Low morale score Two-year pay freeze High staff attrition rate	Simple staff structure Flattish hierarchy Average morale score Stable staff group

metrics programs within improvement initiatives.[1] However, perhaps the most important positive push for metrics programs was the publication of Robert Grady and Deborah Caswell's 1987 book, *Software Metrics: Establishing a Company Wide Program*.[2] The book—and its 1992 revision[3]—describes the Hewlett-Packard software metrics program. Grady and Caswell identify many important, and usually neglected, organizational issues surrounding software measurement. In particular, they proposed, along with other commentators,[3-5] various criteria for achieving a successful metrics program. After studying this and other research, we identified a consensus on requirements for metric program success.

♦ *Incremental implementation.* Implementing a metrics program over time holds significantly less risk than a "big bang" approach.

♦ *Transparency.* The metrics program must be obvious to practitioners. Practitioners must understand what data is being collected, why it is being collected, and how it is being used.

♦ *Usefulness.* The usefulness of metrics data should be obvious to all practitioners. If usefulness is not transparent (or, worse, if the data is not actually useful), practitioners will collect data without enthusiasm and the data will probably lack validity.

♦ *Developer participation.* Developers should participate in designing the metrics program. With high levels of developer participation, buy-in is more likely, as is the implementation of a more incisive metrics program.

♦ *Metrics integrity.* Practitioners should have confidence in the collected data. They should believe it is sensible to collect, accurately collected, and not being "fiddled."

♦ *Feedback.* When practitioners get feedback on the data they collect, it gives them a clear indication that the data is being used rather than going into a black hole. This makes practitioners more likely to view the program positively. Commentators suggest several feedback mechanisms including newsletters, newsgroups, and graphical posters.

♦ *Automated data collection.* This should be done wherever possible. Minimizing extra work for developers also minimizes developer resistance to metrics. It also means that the data collected is more likely to be valid.

♦ *Practitioner training.* Case studies show that a metrics program that has a base of trained practitioners is more likely to be successful. Appropriate training must be targeted at all levels in a company and should range from awareness raising to training in statistical analysis techniques.

♦ *Gurus and champions.* Organizations can increase practitioners' initial enthusiasm by bringing in an external metrics guru (Hewlett-Packard, for example, brought in Barry Boehm; Contel brought in Dieter Rombach). Organizations should also appoint internal metrics champions to help with the difficult and arduous task of sustaining a metrics program.[6]

♦ *Dedicated metrics team.* Responsibility for the metrics program should be assigned to specific individuals.

♦ *Goal-oriented approach.* It is very important that companies collect data for a specific purpose. Usually the data will be for monitoring the attainment of an improvement goal. Unsuccessful or ineffective programs collect data for no specific reason and find they have no use for the data they have collected. The Goal-Question-Metric model[7] has been highly popular among companies implementing a goal-oriented approach to measurement.

On the last two factors, the broader software engineering community does not entirely agree. Although there is consensus on the need for some kind of dedicated metrics team, there is no clear agreement on how

TABLE 2
RELEVANT STAFF AND RESPONSE RATE

Organization	Staff Directly Affected by Metrics Program	Number Responding to Questionnaire			Response Rate (%)
		Total	Manager	Developer	
Embedded systems	24	20	10	10	83
Information systems	125	103	48	55	82

centralized it should be, nor whether metrics should be the team's sole activity. Questions have also been raised about the GQM approach.[8] For example, Hetzel highlights the fact that GQM encourages organizations to use data that is likely to be difficult to collect.[9] This directly contradicts other advice on metrics programs, which encourages organizations to use available or easily collected data.

Notwithstanding these minor doubts, the factors above appear to be commonsense advice on implementing a metrics program. However, these factors are apparently proposed on the basis of either anecdotal evidence or single program experiences.

We spent six months conducting an independent study of these factors at work in two organizations. Here, we present quantitative data that describes practitioner experiences of metrication. This data generally supports—but occasionally contradicts—existing anecdotal evidence about metrics programs.

Although both of our case study programs were structurally similar, one was judged to be successful (by the organization and by us), and the other was judged as not yet successful (by the organization's metrics group and by us). Despite the fact that anecdotal evidence suggests that most metrics programs are unsuccessful, with a few notable exceptions (such as a 1993 study by Ross Jeffery and Mike Berry[10]), only successful programs are typically reported.

Our ultimate aim in this article, which follows on our earlier work,[11] is to present the critical success and failure factors associated with the two metrics programs. In particular, we report on each organization's metrics implementation strategy, quantify the extent to which they implemented various metrics, and detail the ongoing management of the metrics programs.

THE STUDY

Our case study involved two organizations which, for reasons of commercial confidentiality, we refer to as Embedded Systems (ES) and Information Systems (IS). Both organizations

- have more than 10,000 employees, are more than 20 years old, and have complex internal bureaucracies;
- have a large software development function, with more than 400 software staff;
- are fully dependent on in-house software systems to support or enhance their main (nonsoftware) products;
- have a progressive approach toward new development methods and tools;
- have mature quality programs and usually consider the software they produce to be high quality; and
- have had a metrics program for between two and three years.

Some metricating success or failure factors may result from organization-specific issues; Table 1 gives the organizational context of each metrics program.

Phase one: fact finding. At each organization, we started with a few fact-finding interviews with middle managers who had a high level of corporate responsibility for the metrics program. We designed this fact-finding phase to identify the context, framework, and content of the "officially" implemented metrics program. Managers commonly have an inaccurate picture of what is really happening "on the ground." Thus, we designed this phase to establish what state the organization believed its metrics program to be in. In particular, we found out about the organization (such as number of employees in various activities), applications developed, development environment, metrics collected and how they were used, and implementation and management of metrics. We used the results of phase one as a baseline for phase two.

Phase two: data collection. In phase two we wanted to find out what was really happening in the metrics program. We managed to get serious input from nearly all the relevant staff by distributing detailed questionnaires to all managers and developers affected by metrics.

The aims of the questionnaire were to

- identify which metrics were really being collected (as opposed to the ones managers believed were being collected),
- find out how the metrics program was initially implemented and subsequently managed,
- establish the contributions that individuals and groups made to metrics, and

TABLE 3
IMPLEMENTATION FACTORS

Implementation Factors	ES	IS
Consensus recommendations		
Incremental implementation	✓	✘
Well-planned metrics framework	✓	✘
Use of existing metrics materials	✓	✘
Involvement of developers during implementation	✓	✘
Measurement process transparent to developers	✓	✘
Usefulness of metrics data	✓	✘
Feedback to developers	✓	✘
Ensure that the data is seen to have integrity	✓	✘
Measurement data is used and seen to be used	✓	✘
Commitment from project managers secured	✓	✘
Use automated data collection tools	✓	✘
Constantly improving the measurement program	✓	✘
Internal metrics champions used to manage the program	✓	✘
Use of external metrics gurus	✘	✘
Provision of training for practitioner	✘	✘
Other Recommendations		
Implement at a level local to the developers	✓	✘
Implement a central metrics function	✘	✓
Metrics responsibility devolved to the development teams	✓	✘
Incremental determination of the metrics set	✓	✘
Collecting data that is easy to collect	✘	✘

Key: ✓ shows that the recommendation is followed; ✘ shows that it is not

♦ solicit views and experiences of the success and value of metrics.

As Table 2 shows, we had an overall response rate of 83 percent. Thus, almost everyone affected by the metrics programs at both organizations contributed to this study: 24 practitioners at ES and 125 at IS. (ES implemented metrics in only one critical department whereas IS implemented metrics throughout software development.)

Our high response rate can likely be attributed to several factors.

♦ *Topicality.* Although both metrics programs were maturing, neither was long established. Thus, metrics was still a hot issue in both organizations, and practitioners had a lot they wanted to say about it.

♦ *Desire for evaluation and improvement:* In both organizations, metrics program managers fully and publicly supported our study. Because they had not formally evaluated the metrics programs themselves, they were keen to find out their practitioners' views. Developers also seemed keen to contribute, perhaps thinking their contribution might improve the metrics program.

♦ *Independence of the study:* Practitioners were confident that their views could not be used against them by management, and, thus, we suspect, were more open and honest in their contributions.

Because of our high response rate, we're confident that our results are an accurate representation of each organization's metrics program. However, this was not a controlled experiment and does not assess each factor's impact on metrics program success. Moreover, although it would be wrong to generalize results from a case study to the industry as a whole,[12] we believe our results offer many interesting insights that are relevant and applicable to other metricating organizations.

STUDY RESULTS

At both organizations, metric program managers agreed that good metrics implementation was important. Still, as Table 3 shows, they ignored experts' key recommendations discussed above, although ES adhered to more of the proposed success factors than IS did. And, indeed, ES's metrics program was the successful one. Now the details.

Introducing metrics. No comprehensive international survey has quantified the industrial penetration of metrics since the Drummond-Tyler study.[13] Indeed, influential software commentators disagree about the extent to which metrics have penetrated the software industry. For example, Capers Jones claims an encouraging industry-wide use of measures such as function points,[14] while Bill Hetzel is much less optimistic.[9]

Measurement was initially introduced at ES only because they were given external funding to field test the Application of Metrics in Industry (ami) approach to metrication.[1] Ami combines CMM with the Goal-Question-Metric method. With it, you use the SEI CMM questionnaire to establish process maturity, then use GQM to identify metrics that are appropriate for your organization's maturity level.

Although not particularly motivated at the start, ES quickly discovered the value of metrics and implemented the

program reasonably well. Metrics were implemented initially only in one development team, which meant that the metrics program was very close to the developers and could be tightly controlled by the managers.

Metrics implementation at IS was weak. The initial motivation for implementing metrics was that senior managers wanted to monitor productivity. Other studies have shown that weak motivations like this are commonplace, although the relationship between weak motivations and program failure is not clear.

IS set up a centralized metrics group to introduce metrics across the whole development function. There was no discernible use of metrication aids, nor was metrics use piloted. The implementation strategy seemed to consist solely of the metrics group instructing the development departments to start collecting specified metrics.

Developer involvement. Although managers were very involved in metrics design in both organizations, IS had little developer input into the design process. Interestingly, managers at both organizations thought developers were more involved in metrics design than developers said they were: 60 percent of managers—compared to 20 percent of developers—thought metrics design at ES was a joint effort, while 27 percent of managers and 4 percent of developers at IS thought the same.

Goal clarity. We asked practitioners to rank five software development goals according to their organization's priorities and their own. A score of 1 was awarded for the most important goal and 5 for the least important goal. Table 4 shows practitioner perception of the organization's goals. Clearly, neither ES nor IS was good at communicating development goals to their employees.

These poor results are surprising considering that ES was actively using GQM (though the model does not address the method of identifying and disseminating goals). There was, however, much less disagreement when practitioners were asked to rank their own personal goals (and not everyone ranked low costs as their least important goal).

Usefulness. As Table 5 shows, practitioners were generally positive about the usefulness of metrics, although the practitioners at ES were more positive than those at IS (90 percent of ES practitioners and 59 percent of IS prac-

TABLE 4
RANKING SOFTWARE GOALS

Embedded Systems				Information Systems			
Perceived Organizational Goals	Mean Scores			Perceived Organizational Goals	Mean Scores		
	Overall	Managers	Developers		Overall	Managers	Developers
Low costs	2.5	2.2	2.8	User satisfaction	2.6	2.6	2.6
Conformance	2.9	2.6	3.2	Speed	2.9	2.9	2.8
Speed	3.0	3.0	3.0	Reliability	3.0	3.0	3.0
Reliability	3.1	3.6	2.6	Conformance	3.1	3.2	3.1
User satisfaction	3.5	3.6	3.4	Low costs	3.4	3.3	3.5

TABLE 5
USEFULNESS OF METRICS

Metrics Usefulness	Embedded Systems			Information Systems		
	Overall	Managers	Developers	Overall	Managers	Developers
Very useful	30	40	20	15	22	9
Quite useful	60	60	60	44	50	41
Not very useful	10	0	20	25	18	30
Not useful at all	0	0	0	10	9	11
Don't know	0	0	0	7	1	10

TABLE 6
METRICS EFFICACY AND INTEGRITY

	Embedded Systems			Information Systems		
	Overall (%)	Managers (%)	Developers (%)	Overall (%)	Managers (%)	Developers (%)
1. *Is the data collected accurately?*						
Accurate	40	60	20	18	31	8
Not accurate	10	10	10	41	39	43
Don't know	50	30	70	41	29	50
2. *Metrics feedback?*						
Feedback is provided	30	50	10	18	25	11
Feedback is not provided	15	20	10	53	56	50
Don't know	55	30	80	29	19	39
3. *Is the right data collected?*						
Yes	40	60	20	11	13	10
No	10	0	20	46	48	44
Don't know	50	40	60	43	39	44
4. *Is the data manipulated?*						
Often	20	20	20	27	29	26
Occasionally	40	30	50	50	58	41
Never	5	10	0	1	0	2
Don't know	35	40	30	22	13	31
5. *Who manipulates the data?*						
Developers	5	0	10	15	21	9
Managers	35	40	30	40	42	39
Neither	5	10	0	2	2	2
Both	20	20	20	22	24	20
Don't know	35	30	40	20	10	30

titioners thought metrics were very useful or quite useful).

Table 6 shows practitioner confidence in each organization's metrics program. Once again, the metrics program at ES was received much more positively than the metrics program at IS. However, as the table also shows, most practitioners at both organizations were convinced that the metrics data was manipulated. Follow-up research suggests the practitioners were convinced that data was "massaged" (usually by managers) to make a situation look better than it actually was.

We also found a clear difference in metrics feedback between ES and IS. At ES, 30 percent of practitioners said feedback was provided, compared with only 18 percent at IS. Still, uncertainty reigned: 55 percent of ES practitioners and 29 percent of those at IS said they didn't know if their organization provided feedback or not.

Overall results. The approach adopted by each organization gives us a particular insight into GQM, which forces an incremental approach to identifying a metrics set. GQM thus complimented ES's incremental implementation strategy and helped it avoid problems, such as lack of transparency and developer buy-in.

IS used neither GQM nor data that was easily available. They had many problems with their metrics program. IS's approach to identifying a metrics set lacked a clear strategy. Whether GQM is being used or not, an organization must provide its practitioners a clear use for the data they collect. IS did not do this. Further, there must be a clear link between the metrics and improvement goals. If this relationship is not obvious, practitioners lack motivation to put effort into metrics collection. The validity of the data collected is thus compromised.

Our results also gave us interesting insight into the nature of metrics feedback. Practitioners at ES were reasonably satisfied with metrics feedback whereas practitioners at IS were not at all satisfied. At ES, only 15 percent said they did not get feedback, as opposed to 53 percent of IS respondents. This surprised program managers at both organizations.

Before our study, IS managers felt confident that formal feedback mechanisms were in place (although they had not properly evaluated the effectiveness of these mechanisms). Since the study, managers at IS have taken steps to improve metrics feedback.

Managers at ES were surprised at how relatively satisfied practitioners seemed to be with feedback, as they had few formal feedback mechanisms. They did, however, use metrics on a day-to-day basis and regularly discussed results informally. ES also operated an open-house policy for metrics data. Thus, although metrics results were not formally distributed or displayed, practitioners did have access to the data. On the other hand, the program may have been an even greater success if ES had provided more formal feedback.

Finally, many practitioners marked

"Don't Know" when asked about aspects of metrication. Even at ES, where the metrics program was well established and apparently successful, many practitioners did not know, for example, whether metrics data was accurate or if the right data was collected. Practitioner ignorance was a weak spot in both programs. However, practitioners had strong views on some aspects of metrication, indicating better communication and higher practitioner interest. On metric usefulness, for example, few practitioners responded with "don't know."

EMERGING CRITERIA

In addition to the expert recommendations discussed so far, there are other, broader ways to assess a metrics program. Among the most important we identified were metrics collection effort, metrics selection, practitioner awareness, and practitioner attitude.

Choosing metrics. Table 7 quantifies the extent to which each organization collected various metrics data. The table lists a subset of the measures contained in the official metrics programs. IS's metrics list included several other measures and was considerably longer than the ES list.

We determined the most frequently used metrics in each organization based upon the activity levels in Table 7. For ES, the core metrics were

- resource estimates,
- lines of code,
- design review data, and
- code complexity data.

For IS, the core metrics were

- resource estimates,

TABLE 7
METRICS PENETRATION LEVELS

	Total				Manager				Developer			
	Question One		Question Two		Question One		Question Two		Question One		Question Two	
	ES	IS	ES	IS	ES	IS	ES	IS	ES	IS	ES	IS
Function points†	-	86	-	15	-	87	-	19	-	85	-	11
Metrics for size estimates*	85	50	40	11	90	57	60	13	80	44	20	9
Metrics for cost estimates*	90	80	30	26	90	85	50	56	90	76	10	7
Metrics for effort estimates*	70	65	35	27	90	77	60	42	50	54	10	13
Analysis inspection data	40	35	10	8	50	41	20	13	30	30	0	4
Design review data	75	39	25	8	80	44	50	10	70	34	0	6
Design effort data	40	31	20	9	50	40	30	11	30	23	10	7
Code interface data	65	16	20	5	60	13	30	4	70	19	10	6
Code complexity data	70	7	20	0	90	8	30	0	50	6	10	0
Lines of code data	80	43	25	12	100	52	40	19	60	35	10	7
Coding effort data	45	20	10	12	70	49	20	19	20	26	0	6
Code inspection data	30	29	15	7	40	38	20	10	20	22	10	4
Fault rates	25	14	10	2	20	15	10	4	30	13	10	0
Defect densities	20	15	10	3	30	23	20	4	10	9	0	2
Change data	60	28	20	5	70	32	30	6	50	24	10	4
Testing effort data	40	31	10	7	60	49	20	13	20	15	0	2
Test review data	65	25	20	6	70	34	30	11	60	17	10	2

Key: Each entry represents the percentage of respondents answering "yes" to the following questions:
Question One: *Does your organization collect the following metrics data?*
Question Two: *Do you know, from personal involvement, that the following metrics data is collected?*

† = IS practitioners were very keen on function points, so we asked them separately about function point penetration.
* = We asked only about fairly general metrics in order to avoid fragmenting the results.

- function points, and
- lines of code.

Although ES was generally more active in its metrics activity, both organizations favored a typical set of core metrics, dominated by size and effort metrics—primarily used for resource estimation and productivity measurement—rather than quality metrics. This result supports other research and runs contrary to the advice of all commentators. Overemphasis on cost-oriented data is probably another common fault of many metrics programs.

Data cannot be used to motivate productivity if people do not know it is being collected.

Collection effort. There is little explicit discussion in the literature about what constitutes a reasonable overhead for a metrics program, although 7 percent overall has been suggested as an average effort overhead.[15] In our study, neither program appeared to have a large effort overhead. At ES, 90 percent of practitioners said they spent less than 3 percent of their time on metrics-related activity, with the remaining 10 percent spending between 3 and 14 percent. At IS, 79 percent spent less than 3 percent, 16 percent spent between 3 and 14 percent, and 4 percent spent between 14 and 29 percent of their time on metrics activities. The fact that practitioners at ES spent less time collecting metrics data than practitioners at IS was probably because ES used automated tools.

Ironically, IS was not as effective at metricating as ES, and yet spent more effort on its metrics program. Part of this was because IS was collecting more metrics data. It is also likely that some of the IS metrics were not needed or used.

As Table 7 shows, managers at both organizations were more personally involved in metrics collection than developers, although ES had a higher overall participation level than IS. Also, while ES had a successful metrics program, there was minimal developer involvement in data collection. This is because the ES program was deliberately driven by managers and used automated data collection. The metrics program at IS seemed generally inactive, with few managers or developers directly participating in metrics collection. Indeed only 19 percent of IS managers and 7 percent of its developers said they knew from personal involvement that LOC data was collected. (IS's program did not use any special-purpose automated metrics collection tools and was largely paper-based.)

These results suggest that it is important for practitioners to know what is going on in the metrics program, but that personal involvement in collection is not very important. This finding supports other experiential reports[4,16] in which using automated tools and keeping practitioners informed seem like necessary prerequisites to metrics success.

Table 7 also shows that although ES had the most active metrics program, it lacked clarity and goal orientation in two areas.

- *Poor transparency.* Although 70 percent of ES managers said that coding effort data was collected, only 20 percent of developers knew this and no developers claimed to be involved in collecting this data. It is difficult to understand how accurate data can be collected over the life cycle without developer participation (although they may not have realized that time sheets are used for collecting coding effort data). It is also difficult to understand the purpose of collecting such data if developers do not know that it is required. It cannot be used to motivate productivity if people do not know it is being collected.
- *Poor goal-metric coupling.* Although 90 percent of managers said code complexity data was collected, only 50 percent of developers realized this and very few managers or developers were involved in collecting it. The main purpose of collecting complexity data is to control complexity. If developers do not know that complexity data is being collected, they are unlikely to take reducing complexity seriously. It also makes collecting complexity data ineffective.

Practitioner awareness. We analyzed metrics activity in terms of what practitioners knew about the measurement and how involved they were in collecting the data. Our rationale was that, although there may be an *official* metrics program in place, unless practitioners are aware of that program and see themselves as involved in it, the organization has not created the necessary measurement culture and the program is likely to be ineffective. Indeed, many of the metrics cited by both organizations as part of their "official" metrics program were so little known about and used that it is difficult to accept that they were an actual part of the metrics programs.

As Table 7 shows, ES had consistently more metrics activity than IS among both managers and developers. Although the official metrics program at IS contained many measures, awareness of these measures was generally low. Although there was a higher general level of awareness at ES, the awareness gap between managers and developers was lower at IS (an average 13-percent difference between managers' and developers' metrics awareness levels at IS and a 20 percent difference at ES). LOC metrics are a good illustration of this awareness gap: 100 percent of managers and 60 percent of developers at ES knew that lines of code data was collected (an awareness gap of 40 percentage points) compared with 52 percent of managers and 35 percent of developers at IS (an awareness gap of

17 percentage points).

Generally, practitioners at both organizations exhibited poor awareness of what was happening to review and inspection data. Both organizations used reviews and inspections regularly; the review process can be a rich source of metrics data. As Table 7 shows, the use of this data was not obvious to managers or developers at either organization, thus suggesting that the data was being used suboptimally. This is probably a common metrication weakness and is another example of organizations not heeding the published experiential advice.

Practitioner attitude. One of the most important factors in metrics program success is practitioner attitude: If you fail to generate positive feelings toward the program, you seriously undermine your likelihood of success. As Table 5 shows, in general, ES practitioners were more positive about metrics use than those at IS.

A significant influence on attitude towards metrics was job seniority. In both organizations, managers were much more positive than developers about metrics use, introduction, and management. Furthermore, the more senior managers were, the more enthusiastic they were about using metrics. At IS, for example, 87 percent of senior managers were positive about metrics compared to 72 percent of middle managers and 45 percent of junior analysts and programmers.

However, we also found that developers were more positive about metrics than conventional wisdom has led us to believe (Tom DeMarco suspected this was this case.[5]) It is generally thought, especially by managers, that developers are unenthusiastic about quality mechanisms like metrics and cannot see their value. Our results actually show that 80 percent of ES developers and 50 percent of IS developers were positive about metrics use.

It has been said that when developers are asked their opinion about software quality mechanisms, they say such things are useful, but when they're asked to participate they find many reasons why their work must be exempt. This could explain the positive attitudes toward metrics that we found in this study. If this is the case—and developers are only positive about metrics in the abstract—then organizations need to work toward realizing this positive potential. In any case, the relationship between positive perceptions and negative action warrants more research.

Table 5 supports our view that practitioners' perceptions of metrics are strongly influenced by the reality of their metrics program rather than vice versa. If this were not the case, we would expect a stronger alignment between developer and manager views between the organizations. In fact, the table shows that practitioner perceptions varied significantly, even though within each organization the perception patterns of managers and developers were very similar. This has important implications for managers: it means that what practitioners think about metrics and how they respond to them is within managers' control. Too frequently, managers assume that developers will be negative about metrics *per se*. Our results suggest that developer attitudes are built upon experience.

The integrity of metrics data also seems to have a powerful influence on practitioner attitudes, as Table 6 shows. Managers at both organizations were significantly more convinced than developers that

- the data collected was accurate,
- enough metrics feedback was provided, and
- the right metrics data was collected.

Such a manager/developer perception gap is troublesome. For an effective metrics program, it is important that the metrics data not only has integrity, but that developers believe that it has. Our study has not yet examined the integrity of the ES and IS metrics data, so we do not know whether the data has integrity or not. We do know, however, that many practitioners affected by metrics do not believe that the data has integrity. This perception will probably do more damage than if the data has no integrity but practitioners believe that it does. As Tables 5 and 6 show, ES developers were less negative overall about the integrity of metrics data and more positive about using metrics.

There are two probable explanations for the manager/developer perception gap that we observed. First, managers probably have more access to metrics information and data. So, for example, they probably have more feedback and are probably in a better position to judge data accuracy. Second, managers in both organizations were more actively involved in setting up and managing the metrics programs. Consequently, they are less likely to criticize something they were instrumental in setting up. This gap may be compounded by the fact that no metrics evaluations had taken place at either organization and so managers may have been unaware of the problems our study uncovered.

What practitioners think about metrics and how they respond to them is within the managers' control.

EPILOGUE

These case studies form part of a continuing longitudinal study into the quality and measurement programs in several major UK organizations. As

such, we continue to monitor the ES and IS metrics programs.

Our results were taken very seriously by the metrics managers at IS. Since we conducted the study two years ago, IS has either rectified, or is in the process of rectifying, many of the weaknesses we identified in its metrics programs. The program has radically improved, both because of our findings and because IS has since been taken over by a company with a more established quality and measurement regime. In particular, the IS program has been improved in the following ways:

- Metrics responsibility has been devolved to individual development teams and is thus much more local to developers.
- The centralized metrics group now acts in a more advisory rather than managerial role.
- Managers have made a big effort to improve transparency and feedback within the program.
- The actual metrics set has also been revised and is now smaller and more goal-focused, addressing the basic areas of effort, size, change, defects, and duration.

The metrics program at ES continues to be carefully managed and improved in an incremental way. The metrics set has been refined and some metrics are no longer collected. The program is viewed by ES senior managers as so successful that it is now in the process of being rolled out to all other software teams in the organization.

Neither organization has divulged to us metrics data about software quality. Thus, even at this stage, it is impossible for us to report on how the quality of software produced by ES compares to that produced by IS.

Our study confirmed that the success of a metrics program depends upon a carefully planned implementation strategy. We also found that the consensus "success" factors in the literature are generally correct, but that the advice is not always heeded. Success seems particularly linked to an organization's willingness to

- do background research on other metrics programs and use advice given in the published experiential reports,
- involve developers in metrics program design and inform them on the program's development and progress,
- use an incremental approach to implementation and run a pilot of the metrics program, and
- acknowledge developer concerns about metrics data use.

Also, in our study both metrics programs could have benefited from earlier and regular evaluations.

In contemplating the two programs, we suspect that one of the most important but intangible success factors is the approach and attitude of metrics program managers. The successful program was managed with a tenacious commitment to see metrics work. In contrast, the unsuccessful program seemed half-hearted, despite the fact that it was ambitious and expensive to implement. Indeed, at the outset, the odds were probably stacked against the successful organization: it had a weaker quality framework and lower staff morale. This suggests that organizations implementing metrics not only need to make use of the good practices, but must manage those programs with a certain amount of gusto. ◆

REFERENCES

1. *Metric Users Handbook*, ami Consortium, South Bank Univ., London, 1992.
2. R.B. Grady and D.L. Caswell, *Software Metrics: Establishing a Company-Wide Program*, Prentice Hall, Englewood Cliffs, N.J., 1987.
3. R.B. Grady, *Practical Software Metrics for Project Management and Process Improvement*, Prentice Hall, Englewood Cliffs, N.J., 1992.
4. S.L. Pfleeger, "Lessons Learned in Building a Corporate Metrics Program," *IEEE Software*, May 1993, pp. 67-74.
5. T. DeMarco, *Controlling Software Projects*, Prentice Hall, Englewood Cliffs, N.J., 1982.
6. G. Cox, "Sustaining a Metrics Program in Industry," *Software Reliability and Metrics*, N.E. Fenton and B. Littlewood, eds., Elsevier, New York, 1991, pp. 1-15.
7. V.R. Basili and H.D. Rombach, "The TAME Project: Towards Improvement-Oriented Software Environments," *IEEE Trans. Software Eng.*, Vol. 14, No. 6, 1988, pp. 758-773.
8. R. Bache and M. Neil, "Introducing Metrics into Industry: A Perspective on GQM," *Software Quality, Assurance and Measurement: A Worldwide Perspective*, N.E. Fenton et al., eds., Int'l Thompson Computer Press, London, 1995.
9. W.C. Hetzel, *Making Software Measurement Work: Building an Effective Software Measurement Programme*," QED, Wellesley, Mass., 1993.
10. R. Jefferey and M. Berry, "A Framework for Evaluation and Prediction of Metrics Program Success," *1st Int'l Software Metrics Symp.*, IEEE Computer Soc. Press, Los Alamitos, Calif., 1993, pp. 28-39.
11. T. Hall and N.E. Fenton, "Implementing Software Metrics—The Critical Success Factors," *Software Quality J.*, Jan. 1994, pp. 195-208.
12. B. Kitchenham, L. Pickard, and S.L. Pfleeger, "Case Studies for Method and Tool Evaluation," *IEEE Software*, July 1995, pp. 52-63.
13. E. Drummond-Tyler, *Software Metrics: An International Survey of Industrial Practice*, Esprit 2 Project Metkit Sema Group, South Bank Univ., London, 1989.
14. C. Jones, *Applied Software Measurement*, McGraw-Hill, New York, 1991.
15. D.H. Rombach, V.R. Basili, and R.W. Selby, *Experimental Software Engineering Issues*, Springer Verlag, Berlin, 1993.
16. M.K. Daskalantonakis, "A Practical View of Software Management and Implementation Experiences within Motorola," *IEEE Trans. Software Eng.*, Vol. 18, No. 11, 1992, pp. 998-1009.

Tracy Hall is a senior lecturer in software engineering at the University of Westminster, UK. Her research interests center around quality and measurement in software engineering.

Hall received a BA and MSc from Teesside University and is studying for a doctorate at City University, London.

Norman Fenton is a chartered engineer and a professor of computing science at the Centre for Software Reliability, City University, London. His research interests are in software metrics, safety-critical systems, and formal development methods.

Address questions about this article to Hall at the School of Computer Science, University of Westminster, 115 New Cavendish St., London W1M 8JS, UK, hallt@westminster.ac.uk.