IT METRICS STRATEGIES
Helping Management Measure Software and Processes and their Business Value

february 2001 vol. VII, no. 2

What Makes Your Organization Fast? Metrics and Organizational Learning — Part II, People Issues

by Michael Mah

My article on metrics and organizational learning in last month's ***ITMS*** described how excellent companies leverage ideas and collective thought processes to learn faster than their competitors and to compete more effectively. These sentiments have been promoted by John Chambers of Cisco, Alan Webber of *Fast Company* magazine, and John Perry Barlow (during a recent keynote address at the *Pop!Tech 2000*). I wrote about the following ideas:

- It's not just time to market — it's time to learning.
- None of us is as smart as all of us.
- Company metrics and IT baselines can act as organizational knowledge.
- You can organize a metrics function.
- You can overcome fear of a metrics inquisition.

Two notions leap out from this list. The first is the concept of connectivity — connecting individuals into a collective meta mind that can "think."

Continued on page 2.

Defect Metrics, Inspections, and Testing: Part II

by Michael Mah

In Part 1 of this article, I described defect metrics and the behavior patterns that have become evident through scientific research on software projects (see ***ITMS***, January 2001). Metrics on defect-find rates throughout a project's lifecycle (from the early design phases to deployment) show a buildup to a maximum peak, followed by a gradual decline characterized by a long tail.

This peak followed by a long tail is known as a Rayleigh curve, named after the British mathematician. With metrics behavior like this in hand, advocates of defect inspections argue that the peak of the curve can be controlled in two ways: it can be (1) lowered in magnitude and severity, and (2) shifted to the left, so that mistakes are found sooner rather than later. Why wait until testing to correct

Continued on page 6.

executive summary

This issue continues discussions started in January on organizational learning and defect inspections, specifically focusing on organizational issues and how to deal with emotions and tensions in a high-pressure IT environment. The word metrics implies an objective, sterile view of our working world. The notion of benchmarks seems safe — after all, they're simply charts and graphs that tell us the state of our IT programs. In practice, nothing could be further from the truth.

Measures may indeed be about substance and structure and may seem to have little to do with people's feelings and relationships with their peers, but IT is about solving problems and designing solutions, all of which are very personal in nature. Many software designers consider themselves artisans, and their work is about their code, which they identify with personally. One of the best software designers I ever met was also trained as a classical musician.

Both of the topics in this issue have human-related aspects that, if left unattended, can derail your improvement strategies. These articles identify some of these issues, give you insight into them, and suggest ways to effectively solve them. And if you're skillful at this, you might be able to replicate the kind of success achieved by the company described in the defect inspections article. What began as a process improvement initiative in that specific domain actually snowballed into positive benefits in other dimensions — higher functional throughput, faster schedules, lower costs, and a happier organization.

Lastly, I like to think that metrics highlight both the substantive areas we need to focus on and the potential areas of strain in our internal and external relationships. If you're able to identify and solve the problems that affect your day-to-day work, the net result is an improvement in the quality of your life — and perhaps even leaving work before midnight now and then!

Michael Mah, Editor

CUTTER CONSORTIUM

Continued from page 1.

That is, none of us is as smart as all of us. (Yes, all you *Star Trek* fans, I might indeed be talking about the Borg. Of course, when I was younger, the Borg was a tennis player.)

The second notion is that this meta mind can gain access to incredible amounts of information held in a knowledge database. This is not a database containing ideas from outside metrics "experts," although that would be helpful for other reasons. I'm referring to self-knowledge about processes and project dynamics to enable an organization to learn from *itself*: the organization's own secrets — what makes it excellent and what makes it struggle.

Why is this important? Because companies can't afford to waste time and money on shotgun-style methods for improvement. They need to go right to the heart of the matter like a "smart" weapon. Your first thought might be to strike at the people who are screwing up. Wrong! If you want to torpedo your improvement strategies, then do just that.

What I'm talking about are the choke points in your processes. Everyone has them. Communication is a complex thing. Problems are challenging. Writing software code is hard. People get stuck and make mistakes because things were misunderstood or miscommunicated in the midst of the Internet-speed deadlines. So, now that we're all sinners, let's go about fixing the problems.

Finding the Heroes

Just as there are choke points that act as brakes whenever you get a team of people working together, there are also magical times when things just click. Great things happen; teams gel. Acts of brilliance or just plain hard work result in spectacular wins against incredible odds. How did this happen? Well, for crying out loud, wouldn't it be great if we knew and then told everyone, so they could replicate similar successes and score a touchdown for all of us?

This is the anti-*Dilbert* reference frame for benchmarking and process improvement initiatives. If I addressed these notions using that kind of language, you'd all start thinking, "There goes Mah talking about bureaucratic metrics processes. No thanks, I've got real work to do." When people feel they are wasting time and effort, they get cynical, and rightly so.

Ironically, that's exactly what I'm talking about: not wasting time and effort. Light methods are great, but I'm talking about using *smart* methods, *learned* methods, especially in larger organizations where presence and clout in the marketplace make for expensive blunders and enormous lost opportunities when missteps occur. (Remember my references in the December 2000 ***ITMS*** to the now-defunct Ashton-Tate and the sad story of dBase IV?)

When you have no way of knowing what's broken, you can't fix it, so things keep limping along. And when hardworking staff members kill themselves and pull off heroic deeds against the odds, the act can't be repeated because no one understands how it happened. To make matters worse, organizations today churn staff faster than a Maytag washer. When critical knowledge is held only in an individual's mind or in his or her desk files, the information is gone when that employee leaves. What a waste!

Successful metrics programs are about zeroing in on exactly what needs to be done, learning faster, and getting things done more efficiently. They are about *not* wasting time. Time to market comes from time to learning. Organizations that do not learn will suffer and stay dumb, while those that do will beat them in the marketplace, time and again.

Editorial Office: Clocktower Business Park, 75 South Church Street, Suite 600, Pittsfield, MA 01201, USA. Tel: +1 413 499 0988; Fax: +1 413 447 7322; E-mail: mmah@cutter.com.
Circulation Office: ***IT Metrics Strategies®*** is published 12 times a year by Cutter Information Corp., 37 Broadway, Suite 1, Arlington, MA 02474-5552, USA. For information, contact: Tel: +1 781 641 9876 or, within North America, +1 800 492 1650, Fax: +1 781 648 1950 or, within North America, + 1 800 888 1816, E-mail: service@cutter.com, Web site: www.cutter.com/consortium/.
Editor: Michael Mah. Publisher: Karen Fine Coburn. Group Publisher: Kara Lovering, Tel: +1 781 641 5126, E-mail: klovering@cutter.com. Production Editor: Lori Goldstein, +1 781 641 5112. Subscriptions: $485 per year; $545 outside North America.

Dealing with Surprises

"If you don't want to know the answer, don't ask the question."

I first heard this statement several years ago, and it has stayed with me ever since. I wish I knew who said it first because he or she captured one of the most difficult aspects of IT metrics programs in a single phrase. Things like benchmark statistics, both internal and external, can bring incredible insights that produce good outcomes. But they also present peopleware challenges that result in inhibited learning — most of the time, people do not like to be exposed to news that might be interpreted as bad.

Inquiry and metrics introduce the risk that an organization will have to deal with the mismatch of outcome to expectation — i.e., surprise. Yes, the business world is loaded with surprises, and managers deal with them every day. But what I'm referring to here are situations where the element of surprise might shake an individual's (or group's) perception of identity — whether they are competent or incompetent, good or bad, even whether they are lovable or unlovable.

Some time ago, I was asked to benchmark the IT applications development proficiency of a large, Wall Street brokerage firm. It was a five-month engagement that involved interviews and data collection with the firm's IT development staff. Many of these applications were the heart of equity trading transactions involving hundreds of millions of dollars in the New York Stock Exchange. Others involved complex financial analysis of fixed-income securities and derivatives. We spent weeks on-site, talking with developers, managers, directors, and vice presidents. The objective was to find out just how fast this organization of 400 was running.

Just about three weeks from the delivery of our findings, a new senior vice president/CIO was brought in from Europe to take charge of the entire division. The vice presidents and managing directors I had worked with were uncertain about this new fellow, but our preliminary findings showed good news. This group was running with the wolves: its time to market was easily in the upper industry quartile, and its cost performance was right around the industry norm. System reliability needed some improvement because outages in this area could be very expensive, but generally things were good there, too. The bad news was that the group chronically overpromised, so the internal perception was that the group was always delivering late and with less than expected functionality.

When we presented this information to the new top dog, he was gruff and impolite. When the good news portion of the findings was presented, things got worse, with the new CIO challenging our findings. The attack was unexpected and severe. I left the meeting feeling like I had been ambushed, as did members of my team and the vice presidents who sponsored the study. Everyone left the CIO's office in a state of shock. I wondered why in the world I was in the metrics business.

(What made things even more horrible was that one week before the presentation, the workstation with the analysis had crashed and the server backup had failed. We had to work around the clock to recover and barely did, getting down to Wall Street at the very last minute to present the report.)

Much later, I found out that, in this case, good news was a bad thing. Unbeknownst to me, the new senior vice president was on a mission to downsize staff by as much as 25%. His reward would come from achieving *that* metric. Saying that the IT division was as good as the rest of the industry, with schedules much faster than the norm, would interfere with his goal. If people found out about our report in the midst of the slash of the downsizing knife, all hell could break loose. So we became the enemy, and the report was buried and never presented to the hardworking participants who gave their time for the benchmark interviews.

Risks to Organizational Learning from a Peopleware View

The above story underscores a significant aspect of how to make metrics work for you in the context of organizational learning. If you understand an organization's cultural and behavioral frame around information, how it is used, and when it poses risks and opportunities, then you can make the most of your IT metrics efforts.

Simply knowing these risks can help you prepare for difficult conversations and negotiations that come about when metrics reveal hidden aspects of IT performance, both positive and negative. Getting to the point where measures form the basis for real improvement strategies requires practitioners to know where the booby traps are. Most of the time, the objections are well hidden. You have to become sensitive enough to pick up on the cues and then test your findings with an inquiry of your own to understand what gets in the way of true learning and process improvement.

In *Organizational Learning II: Theory, Method, and Practice* (Addison-Wesley, 1996), authors Chris Argyris and Donald Schon summarize some unspoken rules that develop in organizations:

- Let buried failures lie.
- Keep your view of sensitive issues private; enforce the taboo against public discussion.
- Do not surface and test differences in views on organizational problems.
- Avoid seeing the whole picture; allow maps of the problem to remain scattered, vague, and ambiguous.

The deeper and more fundamental norms, strategies, and assumptions are:

- Protect yourself unilaterally by avoiding direct and indirect personal confrontation and public discussion of sensitive issues that might expose you to blame.
- Protect others unilaterally by avoiding the testing of assumptions where that testing might evoke negative feelings and by keeping others exposed to blame.
- Control the situation and the task by making up your own mind about the problem and acting on your view. Keep your view private, and avoid the public inquiry that might refute it.

These behavioral elements determine whether metrics can be discussed and what will be left alone. Argyris and Schon point out the following about an actual case where an inability to deal with surprises revealed organizational dysfunction that inhibited learning:

> The features of the behavioral world that inhibit organizational inquiry into [problems] also pervade the development process itself. Thus, [managers] used strategies of unilateral control and withholding information to protect themselves against and deal with [managing directors and vice presidents]. Managing directors and vice presidents, with their territoriality and wariness of top management, used similar strategies to protect themselves and resist central control. And both groups sought to avoid confrontation and the negative feelings it might provoke by publicly refraining from testing their assumptions about each other.
>
> Thus, features of the behavioral world … constrained inquiry into their processes. One might say that the behavioral world protected itself from exposure.

Whether you're dealing with IT metrics or metrics of another kind, these people-oriented issues can shape and constrain an organization's pattern of action. An organization's "learning system" can be governed by inhibitory processes that cripple acquisition of knowledge that's viewed as threatening — even so-called good news! And it may not be just the top of the hierarchy that's the problem. Sometimes an organization's culture is so laced with a lack of confidence and insecurity that members throughout its tiers practice fear-based behavioral rules.

What to Do? Make a List and Check It Twice

Overcoming these barriers might seem overwhelming, but it can be done. The first thing you need to do is understand both the spoken and unspoken rules within the organization. The spoken rules are referred to in some organizational learning literature as espoused values. These can often be discovered in places like mission statements, core principles, and the like.

The unspoken rules are more difficult to unearth, but they manifest themselves in how an organization acts. They show up in what

is done and what is not done. Some organizational learning literature refers to this as theories in use. A good way to contrast espoused values with theories in use is to think of "talking the talk" versus "walking the walk."

In some cases, an organization not only talks the talk, but also walks the walk. The extent that these two are in alignment outlines the potential for the organization to learn. When they are way out of alignment, chances are there are massive amounts of organizational dysfunction. That can make metrics and organization learning very difficult (but that's a subject for another issue of ***ITMS***).

When you write some of these things down, (initially, this may be a private undertaking for your own understanding), some clarity may emerge about what happens when metrics — and its associates, good news/bad news — come into the picture. For example, if you were to draw a map on discussions of metrics and defect inspections, issues will emerge like a punch list. (In the second article in this issue, there is a wonderful real-world punch list.)

This punch list is invaluable! Look at it carefully. The issues expressed may be different in various organizations, but common themes will likely emerge. You will want to make three columns: the issue itself, what is spoken, and what is unspoken. This will give you a road map from which you can lay out an action plan. Each item on the list is an opportunity. Rank them in order of importance, and target an action strategy for each that includes an element of driving out fear. You will need a sophisticated understanding of the unspoken rules in order to make this happen. Knowing these unspoken rules will give you the sensitivity and understanding needed to overcome the barriers they pose.

Find a Way to Artfully Bend the Rules

Overcoming the challenges of integrating metrics is a subject more suited to a book than one issue of a newsletter. However, I'd like to share with you some thoughts from the field on common ingredients that successful learning companies seem to share.

Emergence of an internal champion. I describe this individual as the go-to person. (Hi folks — you know who you are.) These are the people you look to in the fourth quarter of the game, with 15 seconds left on the clock. They make things happen, and, often, they have since they were about 10 years old. I should add that life is not easy for these hero types. But they are often a critical catalyst to succeeding with organizational learning and metrics.

Senior management support. It starts at the top. If you have rampant dysfunction here, changing things will be difficult but not impossible. If you moan when you read this, don't worry. Chances are, the next reorganization is around the corner.

Education and training. Learning organizations do some of this in (look out) *the classroom*! If you can tackle issues like benchmarking, defect management, inspection processes, deadline management, negotiation, and relationship management through some kind of educational process (on-site workshops, presentations, etc.), then you can shift the thinking within from a "certainty" stance to a "learning" stance. The latter stance tends to be open to new ideas; the former tends to be closed and thinks it knows the truth with a capital T.

Tell the good news first; publicize early successes. If you have a story to tell that might shift the organization, prove your case with numbers. The saying goes, "Without metrics, you're just another person with an opinion." Some people are doubting Thomases. If you have metrics to make your case, it will save you enormous heartache. Think of it as a court of law — many times, having the most persuasive argument comes down to having the best evidence and the best expert witnesses.

Hang in there and be persistent. Some of what I describe here can be deeply entrenched. Be patient and stay the course. These issues didn't crop up overnight, so take things one day at a time and keep collecting your metrics.

Defect Metrics, Inspections, and Testing: Part II

Continued from page 1.

errors in the design and code that can instead be captured and corrected far earlier in the process?

The way in which most IT projects are managed is what creates the problem in the first place. Most of the time, deadlines are set by someone (or something) outside IT. A project team is assembled. Requirements set in … sort of. The project starts, despite not knowing exactly what the customer wants. The requirements both change and grow, but management keeps the deadline fixed (or so it thinks). This scenario is usually the result of inadequate negotiation skills on the part of IT and business analysts, both of whom feel that to get their way, it must be at the expense of the other.

Most IT developers, feeling disempowered, accept this sorry state as their lot and resign themselves to impossible dates, short staff, and long hours at the office. End users feel misunderstood and vulnerable, in constant fear that the project will crash or simply not do what they need.

IT designers, feeling the time pressure, rush the critical early design phases so they can get something coded sooner. More people brought onto the problem exacerbates things, and mistakes slip through the cracks. The defect curves grow in magnitude. Most errors are not found until testing starts. By then, all the defects that need to be found are embedded in the system. It's too late to not insert them; they're already there, and there is no recourse but to test, test, test.

Stopping the Downward Spiral

Defect inspections are but one of the ways to tackle this problem head-on. Other dimensions of solving the problem are absolutely vital and include project and strategic negotiation, conflict management, dispute resolution, software estimation, benchmarking, and project runaway prevention and recovery. However, these are the subjects of other articles.

What defect inspections are designed to do is intervene early. The dynamics described above back load the process. It's very expensive and incredibly risky to do it this way. Projects are typically ramped up to the maximum number of staff toward the end of a project as the deadline looms. At this point, the money throttle is wide open, pouring out the salaries and overhead of the maxed-out team. On a 100-person project, effort is being expended at 100 person-months per month, or about US $1.1 million. If a 10-month project slips by 5 months (something not uncommon), the price to pay for all this is an extra $5.5 million. And that is the tab for just this one project! Outsourcing is starting to sound appealing to your management.

That hefty sum is a big reason why it is important to lead the way and stop the madness. But to do that, you'll have to overcome resistance born out of the cynicism masterfully expressed in comics like *Dilbert*. People are lost, especially in medium to large organizations, where control of one's personal destiny seems remote. At levels throughout the organization, you'll have to overcome resistance to change.

Last month's ***ITMS*** offered a specific approach to help: defect inspections. If you recall, one organization I described achieved unmistakable results. Code defect rates were reduced by more than 50%, with defects per 1,000 lines of code falling from the 30s to the teens. Production jobs with abnormal ends (ABENDS) fell 57%, while the number of jobs ran increased 10%. They installed more than 1.5 times the number of large projects in a given year. That's quite different than $5.5 million in cost overruns per project. Which metric do you want to report to your management and clients?

What objections will you have to overcome? What might they sound like? If you could hear them, I bet they would be similar to what your staff might say. To give you a glimpse, the rest of this article contains a look inside a West Coast systems software company that spent considerable time and investment on an inspection process. Who the company is remains secret, but the things it describes are exactly what I've heard in dozens of organizations.

Overcoming Hurdles

"I don't have the time."

The number one complaint of the inspection process we developed was the time commitment. There is no doubt that doing inspections takes time, but the payback is immense. Conducting inspections adds approximately 15% to a project schedule at the front end, but it will save up to 40% in total project effort by not reworking defects, and it will shorten the project schedule (see Figure 1). The first thing an organization needs to do while implementing an inspection process is to track metrics. Facts speak with legitimacy and authority.

"It will delay the project."

Because inspections take time, project managers (and those looking over their shoulders) feel that projects will be delayed. Although inspections do take time, and early milestones may be delayed, overall, the project is not delayed.

Project managers conducting early inspections at first felt as if their projects were slipping. "We should be done with the designs. We're not in the coding phase yet; therefore, we must be late." In actuality, the design milestones did slip and the requirements and design phases took longer; however, the coding phase went faster than expected. All of the testing efforts were dramatically reduced, and the tests sailed right on through. In the old days, the bulk of a project's effort was spent on the testing phase; by doing inspections, the effort on the front end saved time on the back end of the project. Metrics proved this. Not having time and effort statistics would have made it impossible to make this case.

"Inspections don't work."

When isolated incidents of noneffective inspections occurred, it usually resulted from the wrong people taking part in the inspections. For example, a director anxiously waited for a requirements document to be inspected. After the inspection, the director

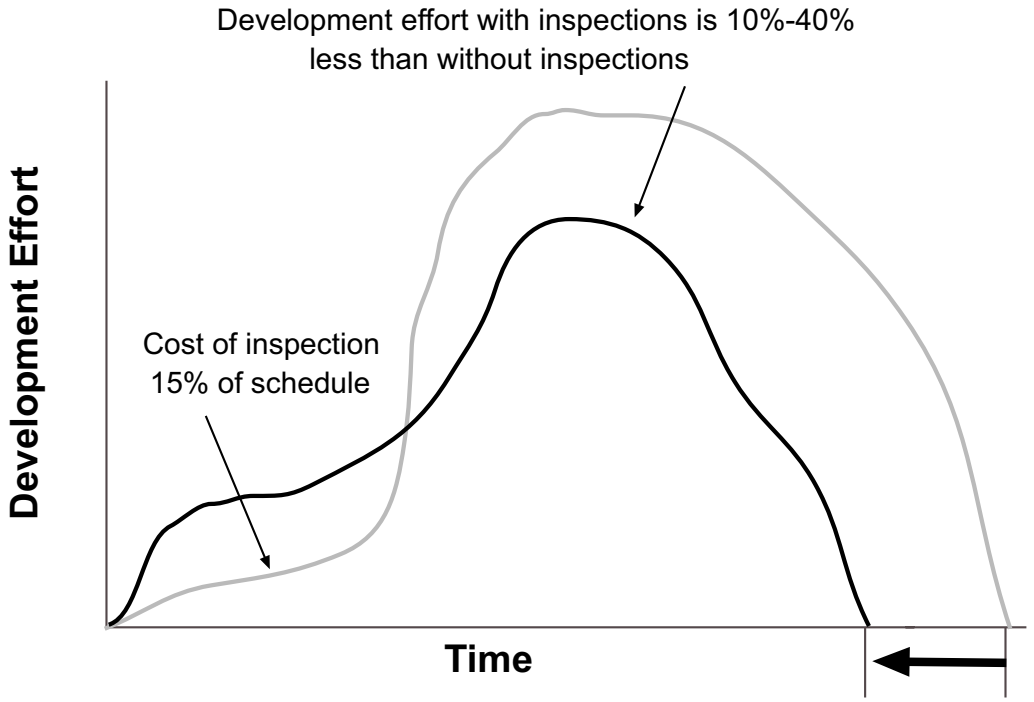

Figure 1 — Lifecycle time and effort reductions with inspections.

commented that the requirements document was not any better and stated that the inspection did not work. Upon further review, the inspection team was composed of individuals who had no vested interest in that requirements document and added nothing to the inspection. It was also noted that vague and ambiguous terms were not marked as defects because "people will know what it means." Inspections work, and rarely have they been found to be a waste of time.

"Doing an inspection is just another administrative task on the project plan."

Project leaders are asked to perform many roles. As the inspection process was being rolled out, project leaders squawked at yet another administrative task to do during the project. A few protested when they saw how the early parts of the project timeline would shift to the right by adding inspections. Some gave inspections the blind faith necessary to add the time to the project plan, with the hope of reducing defects, testing, and (ultimately) the final project completion date. When those projects that incorporated the inspection process completed either on time or ahead of schedule, inspections then started to be added in as an integral part of the process. People would allow time for inspections without actually listing them on the project timeline. A project task of "coding" included both the time to code and the time to inspect.

"My business analysts don't want to inspect."

The business analysts were the most vocal advocates for the inspection process; they attended the inspection training classes from the beginning. During every class, business analysts stated that since such value and benefit was derived from inspections, they would travel across the country if necessary to attend an inspection.

IT members were surprised to hear that their business analysts were interested in taking part. "We thought you would be too busy," IT would say. The business analysts replied, "We're busy doing other things because we don't hear from you. This is so important to us that we'd be willing to travel to attend, because if the spec is not right, we hear it from our end users for months!" For some business analysts, it did take a little convincing and planning on their end to attend an inspection. In these cases, metrics on process benefits were very handy.

"Senior management does not mandate it, so I won't do it."

Once training began, either the president or CIO kicked off each class, demonstrating senior management's commitment to the process and the importance of the staff attending the class. This lasted for about a year. Once it was clear that senior management supported the process, kickoff speakers became vice presidents and directors whose areas used and benefited from the inspection process. Today, the CIO expects teams to use best practices, and inspections are considered best practice. Management backing was critical to its success.

"Inspections are not officially documented in our methodology."

This hurdle was easy to overcome. Since a rewrite of our product development process (PDP) was occurring at the same time as the inspection implementation, we simply added the process and vocabulary of inspections to the PDP. Inspections became officially documented in the methodology.

"My whole team has not been trained yet."

At the time of this writing, the organization had conducted 23 training classes, attended by nearly 400 people out of a total of 1,500 in the organization. You cannot moderate an inspection without attending the class; however, people filling the other roles can take part in the inspection even though they have not been through the training. The teams have developed quick reference sheets that discuss the roles in the inspections and what to expect during the inspection. The sheets are distributed by the moderator to the participants ahead of time. This allows teams to conduct inspections even though not everyone had been trained.

"People come unprepared to the inspection."

Cancel the inspection! Preparation is key to it being effective. If all the participants are not prepared, then it's best to postpone it. Otherwise, the inspection will not be as effective as possible, wasting the time of those who did prepare. Moderators only

needed to cancel one inspection to set the standard. The rest of the participants then realized that they must come prepared.

"There are not enough moderators."

One team availed its members to become company-wide moderators. Moderating an inspection is a skill that improves with each session, but there was still a learning curve for new moderators. This team helped address the learning curve by providing experienced moderators and a breeding ground for new moderators. The benefit to them was increased corporate knowledge and in-depth, detailed information about key projects, which made them more marketable internally. (Most of these team members have since been promoted.) They addressed the concern over who was going to moderate.

"It is unrealistic to inspect all deliverables."

This hurdle is similar to the first hurdle listed. Metrics on time benefit can diffuse this objection and help make the case. The largest benefit comes from inspecting the specification. Since all other deliverables are derived from that, it's in the project's best interest to make sure, at a minimum, that the specification (or requirements document) is inspected.

When even that is not possible, teams are asked to inspect the portion of the project that they are the most confident in. If defects are found in those portions, they may reconsider their confidence in the rest of the project. An alternative is to inspect the riskiest portion of the project. An inspection will at least clean up the portion of the project that is expected to provide the greatest risk.

"The metrics may be used against me."

This is a management issue. All metrics collected need to remain strictly confidential as to who provided them. Metrics are never used for performance evaluations or to determine the efficiency of a programmer. Metrics are shared when they could not be pinpointed to an individual or even a small team. Otherwise, the staff would not report any metrics, or they'd report false metrics. Once trust is broken, it cannot be restored.

Be careful as to how inspections are rewarded. If a company says it will reward the team that finds the most defects, the defect rate may go sky high. The opposite occurs when the reward goes to the team that finds the fewest defects. The best inspection metric is the inspection efficiency or effectiveness. This is a percentage derived by counting the number of defects found by inspections versus the number of defects found in test. For example, if a team finds 6 defects during the inspections, and 4 more are uncovered during testing, the inspection efficiency or effectiveness is 60%. The team found 6 out of 10 defects during the inspection.

"I don't make mistakes."

Invariably, there will be a team member or application owner who is so knowledgeable in their process and applications that they feel they don't make any mistakes. Or, that if a mistake is made, it can be fixed in minutes, not hours. For those teams, the best response is to track the metrics. If the metrics show that by doing inspections the team is not saving any time, then inspections should be stopped.

"This too shall pass."

Many initiatives come and go, and the inspection process was initially greeted with some skepticism as another "flavor of the month," "quick fix," "silver bullet," and "this too shall pass." Adoption of an inspection methodology takes time to filter through an organization. One vice president admitted that when the inspection process was first implemented he laid low, hoping it would pass. A year later, people were still using the process. Only then did he jump on the bandwagon.

"Three days for training! Are you nuts?"

Inspection training requires three days. Application teams were encouraged to attend together; however, having an entire team in training for three days was usually not acceptable. Broken down, the training is one day of lecture and two days of inspections. Thus, the training may be three days, but two of those days involve doing actual inspections on current documents.

Early Success

The inspection process was so powerful that even limited efforts produced dramatic results. IT and business analysts participated, with both finding benefit. The business analysts started to have more productive meetings with IT, and IT started to collect the metrics. Several high-profile projects were piloted using the inspection process. IT was able to draw exact comparisons on projects that used the inspection process versus those that did not. Slowly, the pilot teams added more inspection steps into their walkthroughs. Inspection vocabulary words started to be sprinkled into conversations. The IT test team started to request that inspections be conducted prior to coming to the test team.

The best proponents of the inspection process were those who were initially opposed to it. The converts tended to be very vocal and helped promote the benefits of doing inspections after they were won over. The more people who talked about inspections helped integrate them.

Project A Versus Project B

Two similar projects occurred very early on during implementation using the same platforms and applications. Project A was a project with no requirements gathering workshops, no formal estimations, and no inspections. The project was very large in scope, and the requirements kept changing right up to the install date, which was determined by the business. The end result was 110 production tickets (defects) opened within 24 hours after the install. Normal is 5-15 tickets. A vice president estimated that it would take nine months and several million dollars to fix all those defects.

Right on the heels of Project A was Project B, which started to head down the same path. At this point, the team working on the inspection process implementation stepped in and suggested the following:

- **Use a facilitated workshop to gather the user requirements.** The requirements document was written within one week after the workshop, since all the requirements were basically on the flip chart papers from the workshop. The team leader said it was the most complete and comprehensive spec she had ever written. In contrast, Project A's requirements document took six months to create.
- **Use a formal estimation model to determine the end date based on the work effort of the requirements.** Project B started in December with a hoped-for deadline of June. The estimation model determined the end date to be January of the following year! Since the end date could not change, functionality was successfully reduced with enough early warning.
- **Follow the PDP and inspect each piece of the project.** Each major application involved in Project B sent their teams through the inspection training and conducted inspections throughout the project.

Project B completed its UAT on schedule with no defects, and the project finished two weeks early. It was the first "quiet" install for a project of that magnitude. There was only 1 production ticket opened as a result of the project. The moral of the story is 110 versus 1!

Ongoing Success of Inspections

Data Warehouse Project

The data warehouse was a very high visibility project. Senior management was pushing hard for the project to be completed. Data warehouses — key for future growth and necessary for major organizations — are used to make critical sales and marketing decisions. The director in charge did not want to cut any corners. He approached the methodology team and asked for help with the PDP and inspections. He wanted to use all the processes properly. The data warehouse team used outside vendors and consultants to build the best data warehouse in the industry. The requirements document for the data warehouse was carefully created, taking many months. Special PDP and inspection training classes were provided to all the key members of the IT team, as well as to the business analysts, consultants, and vendors.

During the inspection training, the data warehouse team members brought the spec they had just completed into the class on velvet pillows to inspect! Since the main purpose of the training was to inspect the requirements, the team divided the document into six sections, each with a team of four. Three of the teams inspected overlapping sections to ensure nothing was missed.

After 250 defects, the inspection was stopped! It was clear that the spec and the end result that senior management was looking for were on divergent paths. The director estimated that without doing the inspection, it would have taken a year to figure out they were on the wrong path and another two years to turn it around. The net savings by inspecting the requirements was three years. The requirements had to be rewritten, but due to the defects uncovered during the inspections, the remaining phases went faster than expected. The first phase of the data warehouse was ultimately installed on time. As a result of the data warehouse project, better decisions are now being made to support the business.

Project C Versus Project D

Each of the major business units had undergone complete systems rewrites of their core applications. Project D was a sweeping change to one side of a core system. It was originally planned to take one year, with nine months of that for design and code work and three months to test. It actually took three years, two of which were in testing. Once installed, postproduction issues continued for many months. It was designed, coded, and installed prior to the updated PDP and inspection methodology.

Project C was a sweeping change on the other side of the core system. The project had many failed starts and stops over five years. A new manager took over at the same time the PDP was being updated and inspections were being implemented. The manager saw the benefits that inspections could bring. He was insistent that everything was to be inspected and made sure that inspections were built into the project plan, allowing a month to inspect. Along the way, the project team inspected each deliverable — the spec, designs, test plans, and code. In all, about 35 separate inspections took place.

The manager was giving inspections blind faith. He often said, "The proof is in the pudding. When I get to test, it better go smoothly." As Project C moved along, defects were removed, and the schedule began to shorten due to less rework. The testing phase was able to be reduced by nine months. The project was running very smoothly. By the time it completed integration testing, very few defects were found, and user acceptance testing found zero defects!

This was unprecedented for a project of this size and complexity. It was quietly installed on time last fall. The original deadline was the following summer, nine months later. There were zero postproduction defects, and the manager was promoted to director. The actuals for Project C were one year to design and six months to test, compared to two years of testing for Project D.

Apples to Apples to Apples

Figure 2 shows three projects completed recently. All three projects were roughly the same size (Project 1 was slightly smaller), across the same applications in the same environment with the same complexity. The only major difference in these projects was the team leader and the decision whether or not to inspect. Projects 1 and 2 did not inspect. Project 3 did inspections throughout the project.

Project 1 planned for 32 business days of testing, but actually took 34 days, an increase of 6%. Project 2 planned for 15 business days of testing, but actually took 22 days, an increase of 47%. Project 3 originally planned for 14 business days of testing, but only took 8 business days, saving 43%.

Project 1 had nearly 20% of its test cases fail and had to rework 19 defects (average time to rework a defect was 39.3 hours per defect). Project 2 had nearly one-third of its test cases fail and had to rework 50 defects. That's nearly 2,000 staff-hours of rework, or one full staff-year! Project 3 had 99% of its test cases pass and only had to rework six defects. Project 3 even conducted more complicated test cases by combining issues

Inspections Results

	Project 1	Project 2	Project 3
Days Planned for Testing	32	15	14
Days of Actual Testing	34 (+6%)	22 (+47%)	8 (-43%)
Number of Test Cases	48	304	183
Percentage of Test Cases Passed	81%	68%	99%
Number of Defects Found in Test	19	50	6
Inspections	No	No	Yes

Figure 2 — Test case success rate with and without inspections.

that Project 2 ran as separate test cases. Even if Project 3 were to have double the number of test cases and double the number of defects found, it still would have been less than Project 2.

Inspection Results Reported to Date

In an IT shop with a staff of 1,500, it has been difficult to track all inspections. IT has set up an inspection Web site for teams to post their inspections results; however, not every inspection that takes place gets reported.

The actual number of inspections conducted is estimated to be five times the number that has been reported. Several teams have commented that they conduct inspections twice a week as a normal part of their jobs. They don't post the results of the inspections because they don't post other aspects of their jobs on a Web site. The good news is that in this organization inspections are more widespread than the metrics suggest. However, without the metrics, the progress the organization is making is difficult to track.

Recent Statistics

This organization conducted 23 training classes, attended by 385 IT staffers and 92 business analysts. The number of inspections reported is 469 (5 times more were estimated to have taken place). Of these, 149 were code inspections. And, 5,584 total defects have been removed from all work products, which include the requirements spec, high-level designs, detail technical designs, test plans, and code. Of those defects, 2,599 were operational; 2,985 were minor.

Of the inspections reported, 149 have been code inspections. The operational defects found in code are considered "real." If the code defects were not found during the inspections, they would certainly have shown up either in test or in production. Both propositions are very expensive and time-consuming to fix. In 1998, James Martin Group found production errors to be 40-1,000 times more expensive than requirements errors.

A total of 550 operational defects from code were removed; 3,187 hours were spent conducting the inspections and fixing the defects found in those inspections. However, if those inspections did not take place, and each defect was found during testing, it is estimated that they would have spent 21,615 hours finding and fixing those operational defects (550 defects x 39.3 average hours per defect).

It could also be assumed that not every operational defect found during the inspections would have been found during testing. Testing only covers 10% of the code, which means that defects make it into production. Thus, some of those defects would have taken much more time to fix. The 21,615 hours saved could be broken down into 540 staff-weeks, or 10.3 staff-years saved.

The return on investment (ROI) for conducting code inspections is estimated at 7:1 or 700%. This means that 1 hour spent conducting code inspections saved 7 hours in testing. This is the best way around the "I don't have time" hurdle. If a team leader cannot give 2 hours to conduct a code inspection now, how will the leader have the additional 14 hours necessary to fix the defects later? The breakeven point for code inspections is 2.4 defects per inspection. Is it reasonable that a code inspection can find 2.4 defects in 250 lines of code (LOC). Probably.

A total of 2,059 operational defects were removed from specs, high-level designs, detail technical designs, and test plans of the respective projects in 324 separate inspections. The number of staff-hours spent to conduct and fix these defects was 5,545. If these defects were not found via inspection and translated one for one into the code, the time to fix them would be 80,918 staff-hours (2,059 defects x 39.3 average hours per defect)! Another view is 2,023 staff-weeks or 39 staff-years! Even if half the defects were found by other means, we still save about 20 staff-years by conducting inspections. On average, each inspection finds 12 defects.

The ROI on inspections for all documents is 14:1 (1,400%), which means that spending 1 hour inspecting a requirement or design document saves 14 hours in testing! This should also help you get over the "I don't have the time" hurdle. Metrics make a very strong case — you don't have the time *not* to do inspections! The inspections have reduced the costs and time to market, while simultaneously increasing product quality.

In March 1998, our organization's metrics on the code defect rate were 31 defects per 1,000 LOC, which was in line with the Capability Maturity Model (CMM) Level 1 rate of 33 defects per 1,000 LOC. In time, we were conducting more inspections on work products earlier in the development lifecycle, such as the requirements and design documents. More defects were being found in the requirements documents with more efficient inspections.

Assuming the same defect injection rate, the overall ROI for all inspections increased 10% during 1999, meaning more defects were being found. This in turn led to cleaner design documents and better code being written. Since more defects were found in earlier documents, fewer defects were being found in the code. The ROI for code inspections dropped by 25%. Toward the end of 2000, metrics on code defects dropped to 15 per 1,000 LOC, a decrease of nearly 52%!

Figure 3 shows how the cleaner code impacts the production environment. In October 1999, we ran 89,782 production jobs and had 757 ABENDS during the month. As cleaner code was installed into the production environment over the next nine months, the ABENDS rate dropped by 57.8% to 320 ABENDS in June 2000. During that time, the number of jobs increased by 10.3% to 99,016.

The hours that inspections have saved can be translated into dollars (80,918 x $75/hour = $6.06 million); however, these are usually considered "soft dollars" and are subject to scrutiny. The true, bottom-line benefit comes from the redeployment of resources, which results in the same number of resources working on more projects. Each defect prevented or found means that much less rework *and* the continuation of current work. It's this double benefit of not stopping current work and not redoing old work that provides a significant bottom-line impact.

Additional Benefits

Phantom Inspector

The team has a phrase for the synergy it brings to the inspections — "phantom inspector." Questions the team brings up lead to other questions, which lead to defects that might have otherwise been missed. These synergies work as though there are more inspectors inspecting the documents.

Cross Training

Performing inspections is the fastest way to learn about a project. The preparation and questions developed before the inspections, as

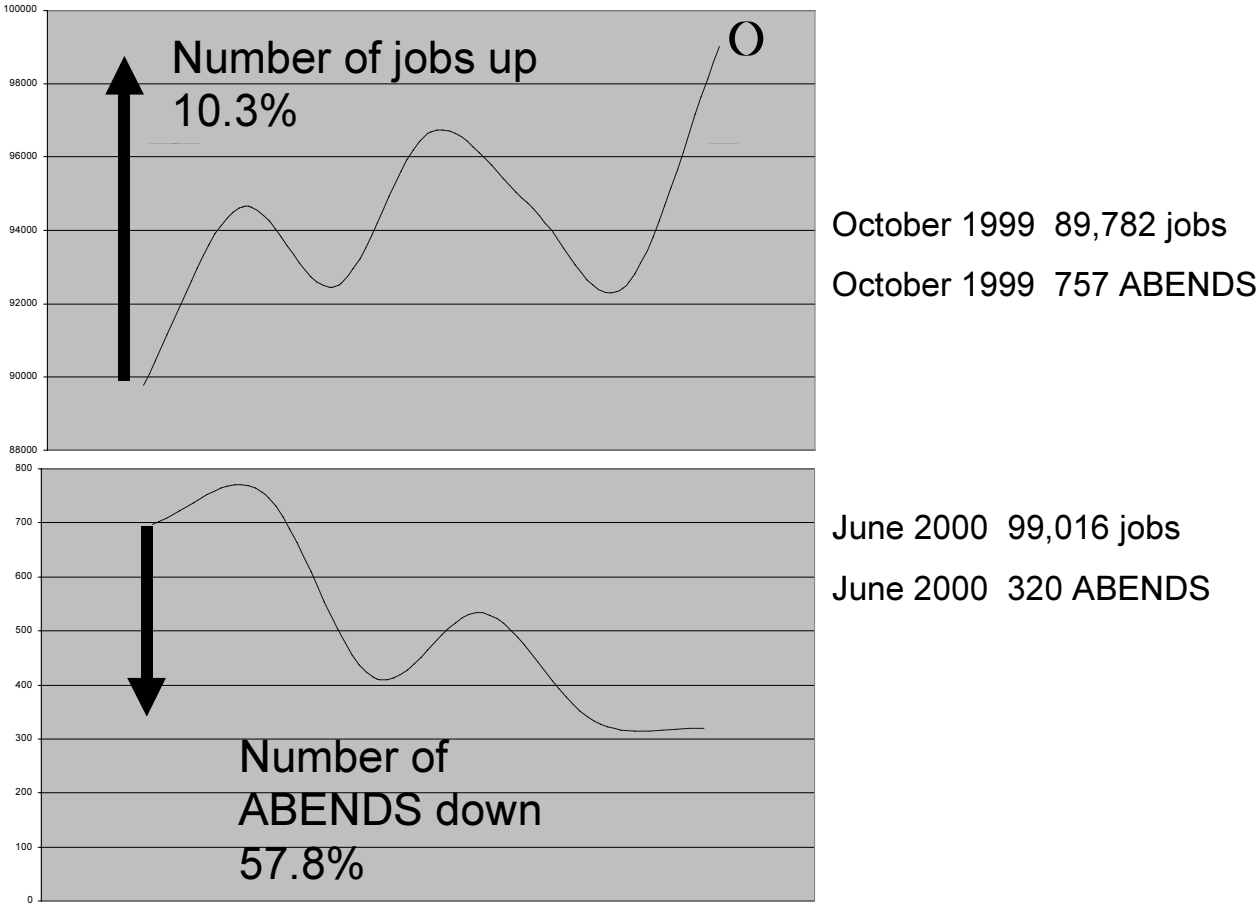

Figure 3 — Improvements in ABENDS metrics.

well as the learning that takes place during the inspections, dramatically shortens the learning curve for any project. When an individual participates in an inspection, especially with vested interest as a moderator, he or she will develop increased company knowledge from the inspection. This has led to career development opportunities and promotions for several experienced moderators. Many good ideas and approaches are shared and observed during the inspections.

Rapid Feedback to Developers

Humans tend to make the same mistakes over and over again. Developers who have their code inspected by others will see their own mistakes. This rapid feedback to developers will help them avoid repeating the same mistakes. The developer will be more careful about the code before bringing it to the next inspection.

Identification and Removal of Systemic Defects

Inspections also find systemic defects in the development process. A systemic defect means that everyone using the process will make the same mistake, resulting in the same defect being injected over and over. Fix it, and that specific defect disappears forever. Part of the inspection process is identifying how each specific defect was injected. Was the defect a misexecution by the author or a systemic defect in the process?

If defects are found to be systemic, the inspection team recommends changes to the development process so others don't repeat the mistake. If one drives the systemic defects out of the process, fewer defects are injected as a result of the process, and the process gets better. All things being equal, the defect injection rate will drop, meaning there will be fewer defects to find. This results in less change of scope, less rework of code, fewer operational defects for the user to find, shorter schedule/cycle time, and

the ability to add more functions in the same cycle time.

This, in turn, makes the development process more capable and reduces the defect injection rate. As Figure 4 shows, to be truly effective, a development methodology must be both capable and usable — i.e., people are able to do their jobs (execute) while using the methodology. Conducting inspections helps development methodologies do both.

Uniform Adoption of Standards

Items are inspected against the exit criteria of that work product. Exit criteria of one phase should match the entrance criteria of the next phase. For inspections to be effective, exit and entrance criteria must exist. Once in place, inspections are conducted against a consistent standard, which ultimately makes systems, applications, and documentation easier to maintain.

What's in It for Me?

What's in it for the average IT staff member? Less overtime, less stress, and more time to work on creative activities. Increased pride in the products developed leads to higher overall morale, which leads to lower turnover. But the largest factor is more time for personal commitments because the pager goes off less often.

Still Not Convinced?

Hopefully, this article has successfully demonstrated the benefits of conducting inspections. In case you're still not convinced, below is an example of two scenarios on a 10,000-LOC program. In the example, the team leader is under the gun to deliver and is not sure that there will be enough time to inspect all 10,000 LOC.

Scenario 1, with no inspections. A CMM Level 1 company will make 33 defects per 1,000 LOC. In 10,000 LOC, there will be 330 defects to find. The testing organization, assuming it can do so by looking at only 10% of the code, will find all 330 defects, one at a time. At 40 hours per defect, the project will spend 13,200 hours in testing (equal to 330 staff-weeks or 6.3 staff-years).

Scenario 2, with inspections. The team leader will plan for and conduct inspections on all 10,000 LOC. It will require 40 separate inspections at 16 staff-hours per inspection. This comes to 640 staff-hours (16 staff-weeks). Assuming that the inspection efficiency is 70%, the inspections will find 231 of the 330 defects. An additional 99 more defects will then need to be tested out, one at a time. This will require 3,960 additional hours in testing (99 defects x 40 hours per defect). The total time for inspecting all 10,000 LOC is 4,600 staff-hours (3,960 + 640), equal to 115 staff-weeks or

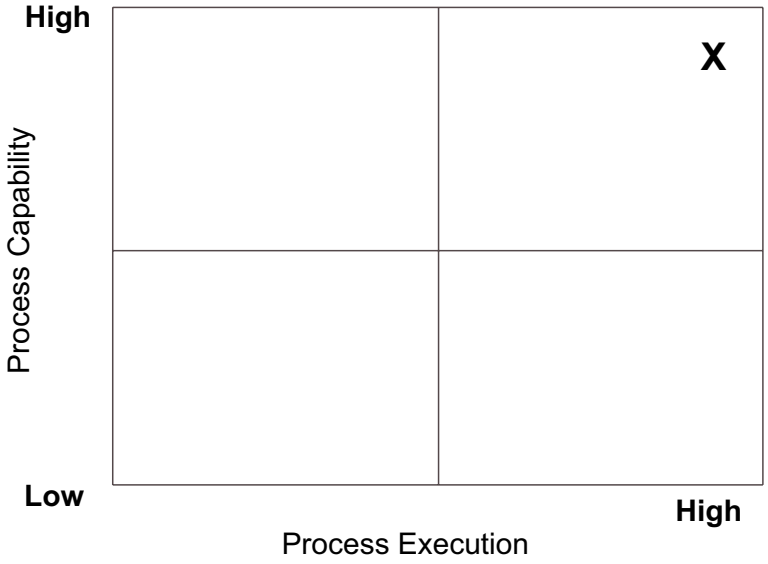

Figure 4 — Process capability versus process execution.

2.2 staff-years. By conducting inspections, the team leader reduced the total testing time by 65%. This includes the time spent doing the inspections.

The above example had a 70% inspection effectiveness rate. Many inspections at our organization have been 100% effective. Also, the example used a rate of 40 hours per defect, the company average. Not every defect will take 40 hours to fix. Some may take only 10 minutes; others could take 80 staff-hours or more to fix. The industry average is 56 hours per defect.

Two key points to remember from the example are: (1) each defect prevented allows current work to continue, and (2) inspections save time on the back end, but there is an up-front cost. The benefits far outweigh the costs, however. At our organization, the minimum cost-to-benefit return has been 7:1, or 700%! As Figure 1 (on page 7) shows, the cost of an inspection will be about 15% of the schedule on the front end. The savings in terms of effort is about 40% when inspections are conducted throughout the project. The schedule compression is also significant; it may be up to 40% as well.

Summary

This real-world example demonstrates how it is possible to make deep and lasting positive changes to your IT organization. It started with an organization taking a learning stance at the highest levels, sparked by the initiative of one person near the middle of the management tier. He became the catalyst for a broader inquiry into exactly how well the organization was performing, sparking a benchmark assessment on the company's applications development and maintenance productivity that would tell it where it stood and why.

This inquiry process gave the company an objective measure of its capability. Knowing its capability yielded some surprises, some of which were unpleasant. The organization discovered both the bottlenecks and the high points of its IT processes. One major finding was the causal factors that were resulting in higher defect rates in its code, which was elongating its delivery schedules because of the enormous time and effort being spent on testing.

The organization was experiencing a back-loaded defect curve, with many defects not being found until after the point of no return, when defects were already deep in the code. By intervening with an inspection process, defects were caught early. The defect curves were reduced dramatically, and the find rate was shifted forward.

This translated to dramatic bottom-line results, sustained and demonstrated by providing legitimacy to the organization's efforts using IT metrics. It was only through its use of measures that the organization found the problem in the first place, made its case to take action, kept the momentum going by showing numbers of its early success, and proved the results in the end.