IT Metrics Strategies

Helping Management Measure Software and Processes and their Business Value

Metrics and the Seven Elements of Negotiation

by Michael Mah

It's hard to open a newspaper these days and not read about the purported economic slowdown. Indeed, *anything* compared to the ferocious growth of the 1990s would suffer. Pressures brought about from these conditions make negotiation very difficult for IT organizations under the gun.

Prior to the slowdown, the Internet-speed mantra "build it faster" had been the order of business for several years. Chaos statistics with rampant schedule overruns have been (and still are) the norm. Demands for ever more functionality exceed rates of delivery. For a majority (perhaps 75% of IT organizations, according to a Cutter Consortium study), backlogs run anywhere from six months to two years. In my view, the economic slowdown means that the other shoe has dropped. Yet again.

This cycle is nothing new. In recent history, we can look at the harsh period of corporate downsizing and cost cutting from 1990-1995. In an April 1998 ***Cutter IT Journal*** article on computer litigation, Cutter Consortium Senior Consultants Tom DeMarco and Tim Lister described the flurry of conflict and litigation that began in the latter half of the 1990s. DeMarco and Lister described embattled IT managers:

> Fearful and under the gun to show improved performance, [managers] fell back on lines like, 'Don't tell me that it can't be done in two years. I am the boss, for God's sake! It will be done in two years!' Engineers, who knew that deadlines were unworkable

Continued on page 2.

Climbing the SEI CMM Makes a Difference on Software Projects

by Stan Rifkin

This article reports on the results of a Software Engineering Institute (SEI) study that looks at the effects of SEI-style software process maturity on duration, effort (cost), and quality. The results are revealing: using the median case, in going from SEI Capability Maturity Model (CMM) Level 1 to Level 2, a typical business application could save 10 months of development time, 75% of development expenses, and 75% of development errors.

Using the database of completed projects established and maintained by QSM, Inc., which contained 2,500 projects at the time this research was conducted, we were able to map SEI CMM software

Continued on page 11.

april 2001 vol. VII, no. 4

executive summary

I recently gave a full-day tutorial at the *Applications of Software Measurement Conference* in San Diego, California, USA. Two statements from audience members spoke volumes on the dilemmas IT managers face: "I'd like to hear more on negotiation and less on metrics"; and "I can't do a thing about my deadlines. I also have little leverage regarding the number of people on my team. So I'm left with a monstrous negotiation problem."

What role do metrics serve in negotiation? How do industry statistics, productivity baselines, commercial metrics estimation tools, and metrics repositories help us solve conflicts within IT? For the past few years, we've been urged to run at Internet speed. As a result, some IT departments resemble train wrecks. Now, with the purported economic slowdown, outsourcing will likely be the way that companies try to cut costs. The double crunch is on.

Negotiations will be very difficult. Not many IT professionals have had formal training in dispute resolution and negotiation for this conflict-intense environment. But effective negotiation, combined with metrics, can create a powerful, winning combination. My article tackles this subject.

Stan Rifkin then gives us metrics statistics that are useful in negotiations around process improvement. If we were to expend the effort and investment to reach Capability Maturity Model (CMM) Level 2, 3, and beyond, what kind of payback might our metrics show? How much speed, cost, and reliability improvement might a typical project exhibit at higher CMM levels? Rifkin's statistics are culled from Software Engineering Institute CMM-level distributions and research data from the QSM industry database (currently comprising more than 5,000 projects).

Finally, Larry Putnam and Ware Myers tackle what you have to manage. They describe the conceptual dimensions of project size, process productivity, time, and effort, and how they are interrelated. If you want to shrink time, then you have to do something in the other dimensions to make it happen. So — go make it happen!

Michael Mah, Editor

CUTTER CONSORTIUM

Continued from page 1.

> or that quality would suffer, just shrugged in the face of such insistence. And so the impossible targets were accepted. They made their way into contracts. And the contracts were signed. The projects tried their best and failed. And then the parties went into court.

The Cycle Repeats Itself - Maybe

If IT professionals want to break this cycle, I believe they can. But it will require a new skill — one that uses negotiation along with IT metrics. Neither can exist effectively without the other. For example, having solid metrics but no negotiation skills can lead to unresolved conflict. In this instance, an IT manager might disregard reliable numbers and cut a bad deal, one that leads to a poor outcome (as described above). Strong negotiation skills without the right metrics might lead to a deal, but it could be flawed because of bad measures, again leading to a bad outcome. The truth is that you need both.

A combination of the two can help parties talk constructively about the right things in order to obtain fair, wise, sustainable outcomes. This is the case whether we're talking about productivity improvement commitments, promises for a certain amount of functionality within a given deadline, negotiated project schedules and budgets, or any combination of these factors.

The fact is that deadlines will not go away (they might even get worse), while costs will come under increasing pressure as the slowdown occurs. Staff resources will either be frozen at current levels or reduced. This sets the stage for even more tension and conflict that will have to be dealt with.

Unfortunately, typical IT personalities have a tendency to resign themselves to fate in this situation and overpromise, much as was the case in the 1990s. IT professionals find it hard to say no to ever-increasing project demands. Some even overpromise in a naive effort to maintain some sort of job security. When budgets get tight, this mindset propagates the belief that management only looks to retain workaholics who never say no and are willing to work long hours with unpaid overtime. In this environment, it's timely to tackle the subject of metrics and negotiation.

Negotiation - Not Something You Typically Learned in College

It's probably safe to say that most IT professionals and their managers have not been formally trained in negotiation. As a degreed electrical engineer, it certainly wasn't in my undergraduate curriculum 20-plus years ago. We dealt with pure science. People issues and the negotiation of conflicts seemed secondary. I became almost obsessed with the field of negotiation at the graduate level, but only after experiencing and observing conflict firsthand in industry for years.

Almost every person I talk to in software, IT, or otherwise, describes negotiation and organizational/people issues as *the* most difficult aspects of their jobs. It's no wonder. We're not trained for it. Even when we are, conflict can be difficult and emotionally taxing.

I recently taught a full-day tutorial at the *Applications of Software Measurement Conference* (*ASM 2001*) on negotiating project scope under mandated deadlines. Although I was very pleased with the responses, I was surprised by comment sheets that said, "Next time, I'd like more discussion on negotiation and less on metrics." The irony was that I had constructed a syllabus with an emphasis on metrics because *ASM 2001* was a metrics conference! But this kind of feedback, plus earlier observations, gave me new insights about what's needed in this technology economy.

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.

Beat the Other Side to a Pulp - Or Use an Alternative

From a sociological perspective, people don't seem to have very good track records when it comes to dealing with differences. We're taught that when conflict occurs, the end game is to win. And to win, you must beat the other side to a pulp. If the other side concedes more than you, they lose and you win. It's as simple as that.

Power, tactics, and leverage are the order of the day in this environment. This pugilistic model of human behavior seems to appear in disputes ranging from two people in a street brawl to battles between departments, lawsuits among corporations, and wars between nations.

In landmark works on negotiation such as *Getting to Yes: Negotiating Agreement Without Giving In*, authors Roger Fisher, Bruce Patton, and William Ury argue that there is a better way, one that is focused on something known as principled negotiation, or negotiation on the merits, developed as part of the Harvard Negotiation Project. This work and others shift negotiation into a different reference frame than what many are accustomed to, articulating what is described as the seven elements of effective negotiation. They are:

1. Evaluate and develop alternatives.
2. Probe for underlying interests.
3. Invent as many options as possible.
4. Use tests for legitimacy.
5. Develop well-planned and realistic commitments.
6. Ensure effective communication.
7. Have the process that builds (not destroys) relationships.

These elements can be compacted into four basic points that provide a proven, reliable framework that anyone can use. In IT, metrics play a vital role in each. These four points are:

1. Separate the people from the problem.
2. Focus on interests, not positions.
3. Invent options for mutual gain.
4. Insist on using objective criteria.

The remainder of this article will illustrate practical approaches of using IT metrics within each domain. We'll describe how to use productivity baselines, metrics and project estimation models, and other metrics concepts within these four basic points to more effectively manage conflict within the world of IT. I hope it reduces the number of runaway projects in your organization and, in the best case, helps a project become successful, as opposed to winding up in a courtroom.

Use Metrics to Separate the People from the Problem

"Without metrics, you're just another person with a different opinion."

— Stephan Leschka, Hewlett-Packard

Let's say you're arguing the all too familiar case of a project that must deliver a certain amount of functionality within a predetermined deadline. Most IT groups try to solve this problem without any benchmark statistics of their own productivity, without any estimation models to test all the possible scenarios, and without any size metrics or historical analogies to negotiate how much they can build within that deadline. In these cases, arguments become ego wars.

In these "he said/she said" conflicts with no data, perceptions become reality. These perceptions can include things like:

- "IT always pads their estimates."
- "Management and marketing always ask for the impossible."
- "It's only a little scope change; there's no reason it can't be included without changing the deadline."

The game becomes scoring points, proving that you are a "team player" if you say yes, building as much leverage as you can to get your way, and digging hard into positions to yield concessions only if you can get something in trade somewhere else.

Without data to deal with substantive aspects of the problem, these perceptions cause people to "lock in." How you see things depends on the side you're on. A deadline looks very different to a development manager than to a marketing vice president. To the former, making a deadline might mean another six months of working nights and weekends; to

the latter, it might mean capturing market share, making revenue targets, and having the rewards that go along with these things. The deadline, and perhaps missing it, means different things depending on one's partisan perception. One example that many of us may care to forget: a dimpled ballot or hanging chad was viewed quite differently, depending if you supported Al Gore or George W. Bush in the recent US presidential election.

Within IT, when problems are tackled without measures to frame the substantive issues, the people issues become enmeshed with solving the problem. Arguments over the problem can degrade into personal attacks and counterattacks. Communication ends up focused on the wrong things, and relationships suffer because cutting a deal uses ego and power tactics instead of objective history with reliable metrics data. What people agree on becomes keyed to how they view their identity. The boss might say, "I am the boss, for God's sake. It *will* be done by the deadline!" The development manager or project leader might be thinking, "I am the superhero mega-competent tech star. I will overpromise and take risks and be a hard-charging team player!" But so far, at least for the past 20 years or so, statistics on overruns and project failures suggest that we're not doing very well with this approach.

Instead, we need to tackle substance. It's amazing that once a few measures are in place, the problem takes on another dimension (provided that your metrics are reliable). Having numbers and metrics charts means you can test whether deadlines or productivity targets are realistic. If things look out of whack compared to demonstrated reality, the conversations might take on a different tone, such as how to better solve the joint problem that you share. Collaboration should be the goal — not competition.

Getting the Right Metrics: Focus on Interests, Not Positions

When I first heard this statement — focus on interests, not positions — I immediately tested it with several of the hard-line positions I've heard echoing through the halls of IT, such as:

- "We want to reduce IT expenditures by 20%."
- "We want X amount of functionality no later than eight months from now."
- "The schedule absolutely cannot slip past September."
- "Head count can be reduced by 5% per year with no drop in productivity."
- "Our goal is Capability Maturity Model Level 5 in five years."

All of these statements are articulations of positions, usually demanding more for less. And for each point, there is often a vociferous argument for its counterpoint, which is intended to reduce this negotiation to a haggling toward a middle ground.

In all these cases, the haggling over positions is about what is known in negotiation circles as distributive bargaining. That is, the parties are looking to cut a pie (distribute the pieces) and concede as little as possible. The goal can be a productivity target, an agreement on project scope, the deadline for the project, or the head count of the team that is assigned to do the work.

Usually who "wins" is dependent on who has the most power and leverage. The default here is assumed to be IT management, not the teams involved in building new systems and technologies. It's assumed that the people creating the technology are disempowered because they are *not* the boss, so they concede, albeit after some resistance.

How to Get to Underlying Interests — What's the Point?

Figuring out the interest that underlies each of these positions might seem tricky at first, but there is an incredible benefit to doing so. Some might question the purpose of asking their antagonist the reasons for his or her position, but there are times that parties can be focused on the wrong problem. "Cut costs" might actually mean "improve profits." If cutting costs means downsizing staff on critical projects, with deadlines being missed or projects being canceled outright, revenues might suffer. If that were to happen, achieving the underlying interest of improving profits is damaged.

Another version of this scenario plays itself out quite frequently: staff is cut, yet the deadline is steadfastly held, without any change in scope. Then the scope grows because people aren't able to negotiate well. When the delivery date arrives, pressure is exerted to get the system deployed. The fact remains that it isn't ready, and the choice narrows to cutting functionality to deliver only a partial system or releasing the whole enchilada with high defects. This choice is usually faced by a team, wearied from working overtime, as a desperate, last-minute measure. If the system is released prematurely, the organization suffers field outages along with high maintenance and production support costs (which will then be reflected in *those* metrics).

In these examples, both parties were intent on cutting the pie and, in doing so, wound up with a poor result where both parties lost — their pie actually shrank because of their actions. An alternative might have been to craft a solution where they could grow the pie, with more for each side as a result.

How to Change the Game — Use Inquiry As a Conversation Model

For the most part, arguing over positions involves parties making statements and declarations about *their* position and vociferously arguing to convince the other. Very little listening occurs as people dig in their heels, and the situation becomes emotionally charged. When there are no metrics to legitimize the respective scenarios, the battle often become personality-driven, with arguments over right and wrong becoming the order of the day.

The way to break this pattern and decipher underlying interests involves using a different mode of communication, one based on dialog that explores both sides through inquiry. One purpose of this mode is to uncover shared interests, rather than focusing on the conflicting ones. Often, these shared interests are hidden because the conflicting ones take center stage. To bring the shared interests to the surface, a modality shift toward gaining understanding of the underlying issues is the key. In the example of the deadline, a development manager might ask, "What are the key elements around the importance of the April delivery date?" or "If we were to make April, which features would be deemed most critical and why?"

"Why?" is an often overlooked tool in acquiring a better understanding of underlying interests and motivations. "Why" questions are not framed in an antagonistic way; they genuinely seek to understand what is accomplished by goals like "cut by 20%," "reach 6 Sigma," "achieve Level 5 in five years," or "deliver in eight months."

In the case of the deadline scenario, inquiring about the key features shifts the dialog toward a shared problem on prioritizing functionality. This might result in critical conversations about requirements from the end user in the context of the date, as well as the staff resources needed to accomplish the goal. In the end, the negotiation might be more about reverse estimation. That is, within a fixed date, which features and in what priority order can they be included within the desired date? "All of them" might not be realistic; metrics would come into play to verify whether this is legitimate. Part of the negotiation might involve a workaround (a Plan B) in case Plan A fails. This is about good risk management — everyone needs a Plan B.

A simple example of shifting from positions to interests is the story of two people arguing over an orange. Both sides want as much of the orange as possible, if not the whole orange. If one person were to ask why, he or she might discover that one wants the rind to make an orange cake, while the other wants to drink the juice. It turns out that both sides would have been willing to discard all the juice or the entire rind because those elements weren't aligning with their interests. If sharing the orange involved a deal where one person got all the rind and the other person got all the juice, both would win, more so than if each side only got half an orange. If they stuck to only arguing the position of wanting the whole orange, they would not have discovered that a better deal existed for both of them.

Thousands of experiments using real cases in negotiation research demonstrate that hardline distributive bargaining (cutting the orange) results in both sides being worse off. In the court case described by DeMarco and Lister, cutting a bad deal based on "I must get the answer that I demand" resulted in

neither having a "done" project. The two parties were left with nothing except a courtroom battle over who was to blame.

Use Metrics, Models, and Benchmarks to Invent Options for Mutual Gain

Creative use of IT metrics shifts the mindset from entrenchment of position into crafting scenarios that may satisfy underlying interests. This overcomes the misperception that there is only one answer for an IT project (mine). In the case of a project with a tight deadline, the negotiation often narrows to a "he said/she said" over the date and whether it is possible or impossible. But there are several options to explore, including adding the right staff at the right time, buying rather than building, best case/worst case project scope, and reliability tradeoffs.

Consider the scenario shown in Figure 1. This is a simulation of a staffing scenario for a project yet to be launched; it was produced by a commercial estimation model. The major phases along with the staffing for each are drawn along the calendar axis, with the deadline shown by the arrow at the right.

The project scope is simply too large to bring in the project at the required deadline. At this point, the stakeholders all have the same problem: they need a scenario that shows an end date finishing to the left of the arrow, not the right. As an analogy, imagine they are sharing a ride to the airport to catch a 4 pm flight. All indications are that they will likely arrive at the gate at 4:15, but they need to get there by 3:30 at the latest to make it on the plane.

Effective negotiation moves toward identifying options to be put on the table. One option might be to add staff to the project, exploring a scenario that builds the team to a peak of 12-15 people instead of 10. The tradeoff is that under-the-gun projects that add staff usually experience more defects. This will have an impact on testing, and there might be a higher risk that quality levels might not be adequate when the system is pressed into service.

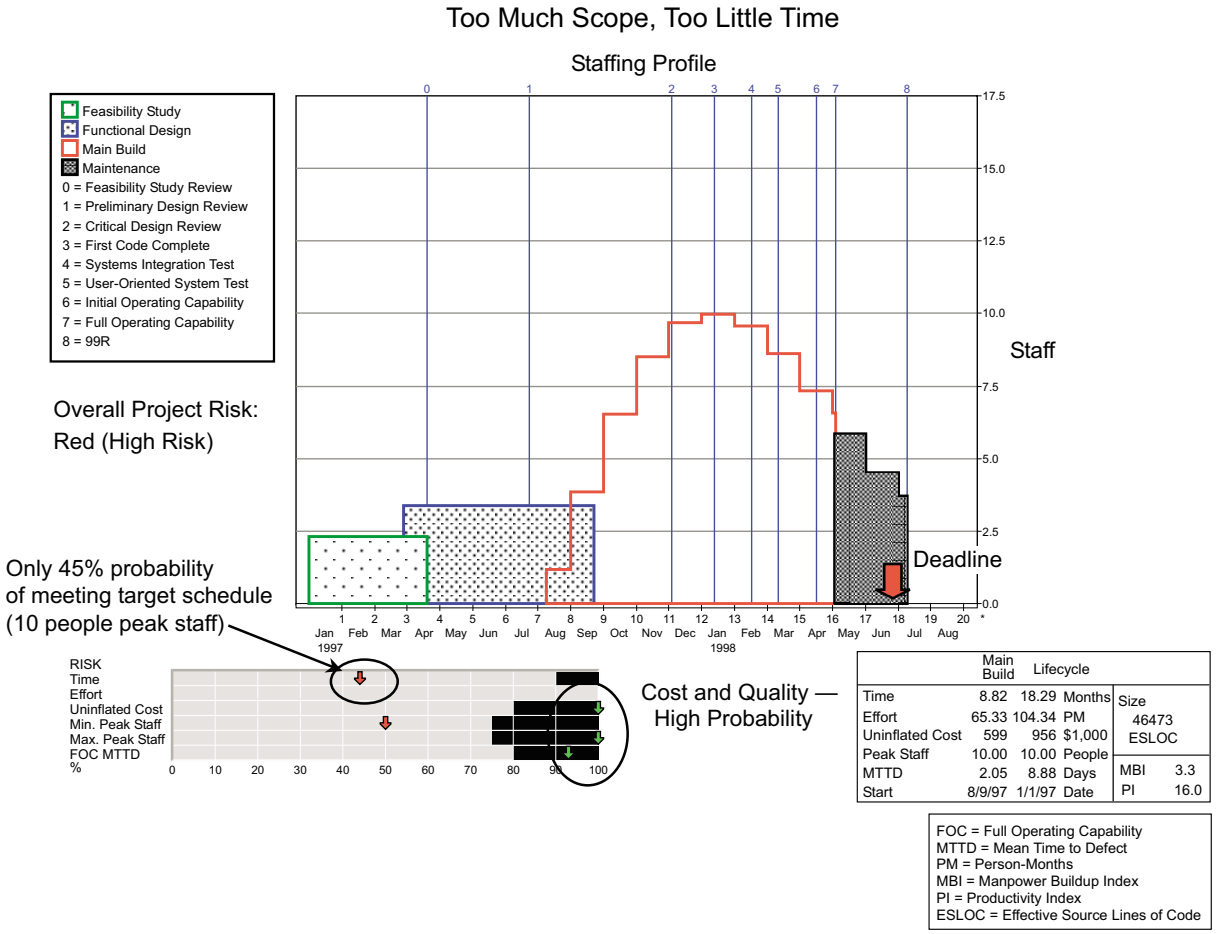

Figure 1 — A project that will miss its deadline: red-light condition.

Another scenario might be to trim the promised functionality to a threshold that makes the date. In that case, the negotiation will focus on feature tradeoffs with marketing or the end users. Although this might be tough to accept, it is better than the alternative. Most would agree that having 28 work requests comprising a certain number of programs that function properly is more desirable than shooting for 36 work requests, many of which might not work.

If one were to examine the feature/deadline tradeoff, a metrics analysis might include finding the line in the sand for project scope. That is, if one were to pull back from overpromising, how much functionality should the team shoot for within the project end date, within the budget, yet still satisfying reliability goals? That might look like Figure 2.

Finally, Figure 3 shows what this option looks like in relation to the deadline. Now, instead of the project starting off in a red-light condition, the commitments are negotiated to the levels where the project has a higher likelihood of success. This option might be in the best joint interests of all the stakeholders. The key will be to resist the temptation to add more to the wish list once the project begins and thereby grow the scope past the cutoff point.

Inventing options takes some time, especially with manual methods. They will take less time when an organization builds its own metrics database and has a productivity baseline in use to reliably estimate new projects. Combined with any of the available commercial estimating models, these options can be generated in minutes, not weeks.

During one such exercise with a *Fortune* 100 communications firm, the options that a team produced during an estimation workshop produced four different scenarios in a matter of 30 minutes. A senior manager that participated said that generating just one option like the ones constructed during the workshop would have taken his team two weeks. In the face of laborious manual methods, it's no wonder organizations generally don't invent multiple options.

Because of this, most negotiations on deadlines (or budgets) become zero-sum games. One month sooner to you means one month less time to do the same amount of work for

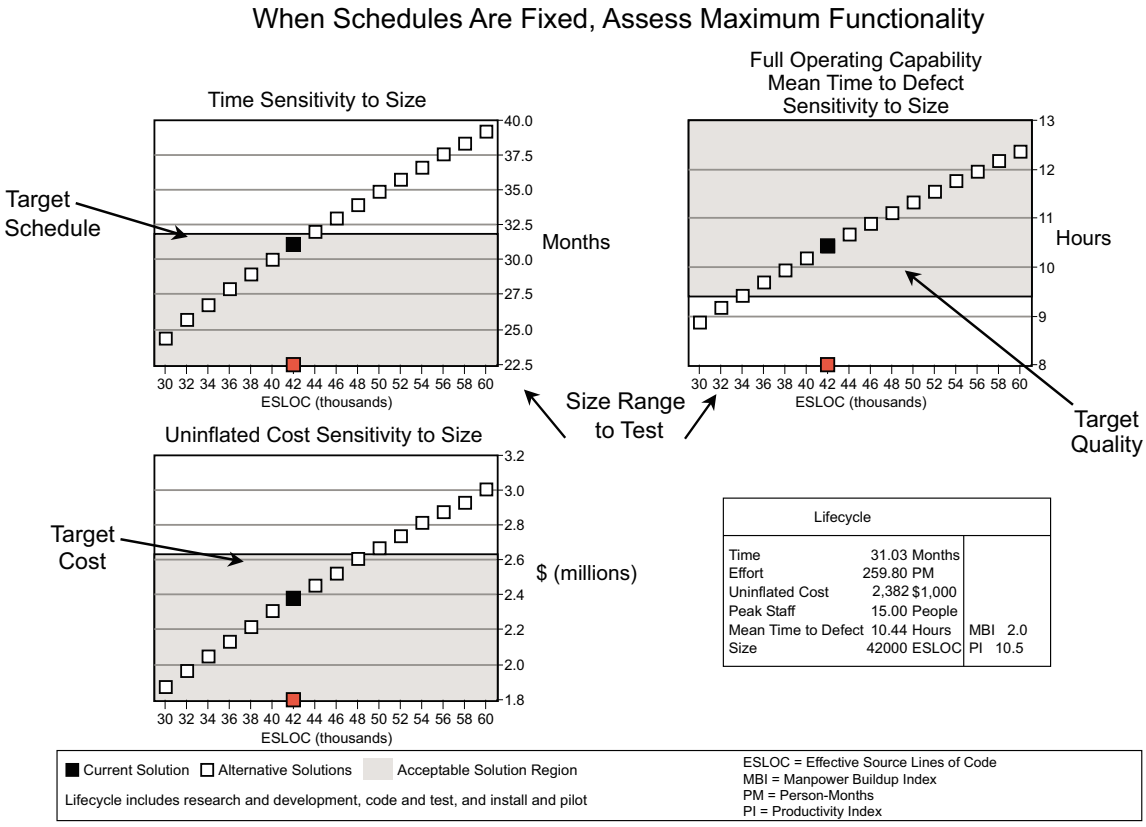

Figure 2 — Sensitivity analysis: feature/schedule tradeoff.

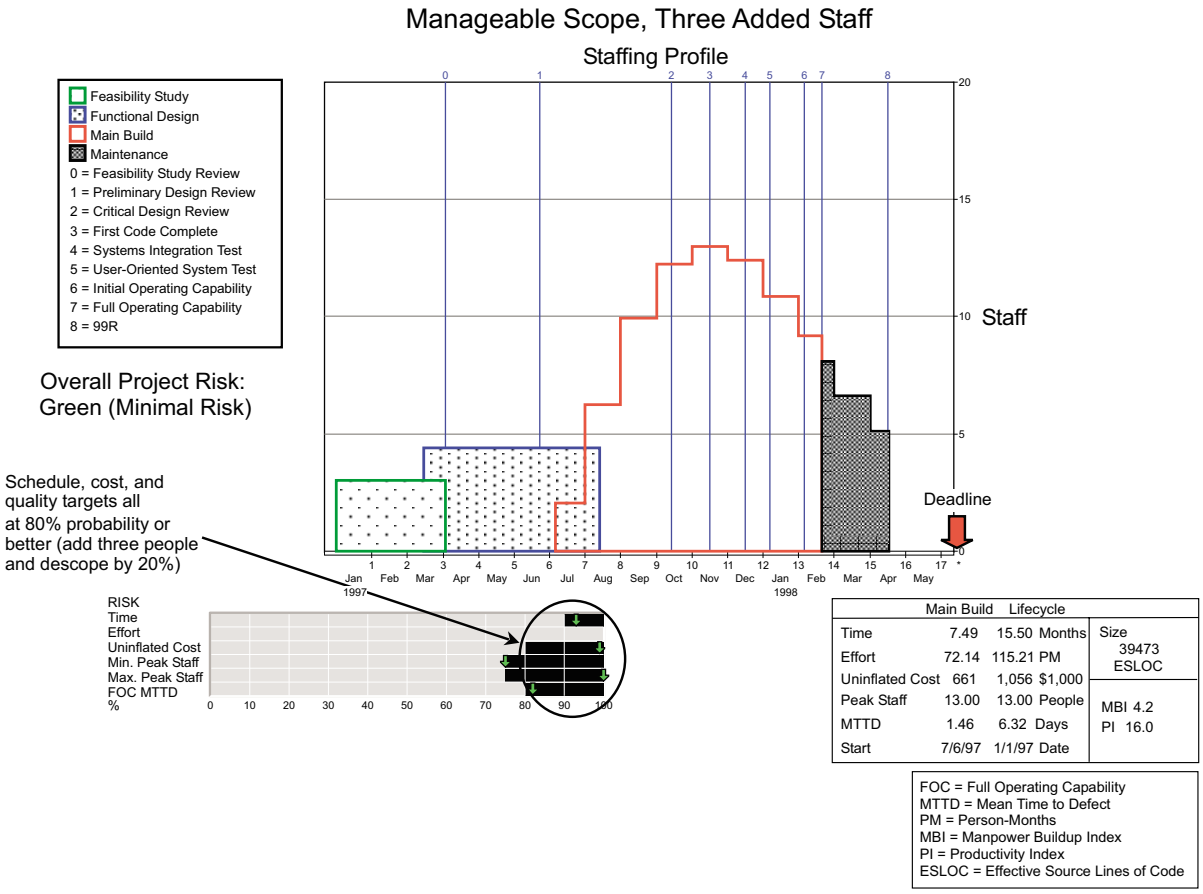

Figure 3 — A project that will make its deadline: green-light condition.

me, and the game begins. Creatively finding a solution doesn't happen when options are narrowed.

Insist on Metrics Benchmarks As Objective Criteria

Objective criteria (such as IT metrics) help people decide what is fair, based on merits of the problem. Negotiation on the merits seeks to validate the terms of an agreement against a standard, such as a benchmark. Without objective criteria, agreements are often determined by which party has more power over the other. As Fisher, Patton, and Ury write:

> Trying to reconcile differences on the basis of will has serious costs. No negotiation is likely to be efficient or amicable if you pit your will against others, and either you will have to back down or they do. And whether you are choosing a place to eat, organizing a business, or negotiating the custody of a child, you are unlikely to reach a wise agreement as judged by any objective standard if you take no such standard into account.

What counts as a standard to assess questions like, "Is eight months fair?" or "Is US $2.5 million a realistic budget?" In the case of a $250,000 asking price for a house, the standard would be the recent selling prices of similar homes in the neighborhood within the last six months. If you had a goal for your cholesterol level at your next doctor's appointment, would 180 be attainable given your personal and family history? How did your last four checkups compare against the population average of 200? In the words of Edward Tufte of Yale University, quantitative methods seek to answer the question, "Compared to what?"

IT metrics and benchmarks help answer this question. Consider the chart in Figure 4. It shows data points for schedules of three projects against an organization's trend derived from its own IT benchmark data collected during the previous year. Smaller

projects are on the left; larger projects are positioned toward the right. The two data points shown in the squares with an X in them are finished projects. The third, a solid square, is an estimate.

Reading this graph, the findings would be that the 960 project took slightly longer than the historical average denoted by the center line. The 8240 project was quite a bit faster than the average, as it plots a further distance below the center line. The third data point (solid square), a deadline for a project that has not yet started (called SmartMed), plots in a position totally inconsistent with the two recently completed projects. Moreover, it is quite far below the norm denoted by the center line. This benchmarking of the deadline is, in essence, "sanity checking" the target date. From this, one might deduce that the deadline for SmartMed is quite risky. It stretches the limits of being fair to the team. It would be considered even more risky if the SmartMed project were deemed more complex than either of the previous two — something that is not uncommon.

If this were a proposal to a client organization by an outsourcing supplier, it would behoove the customer to request more metrics data to assess whether the contractor was promising something overly ambitious. That way, the vendor's SmartMed proposal could be assessed against the outsourcer's previous performance, much the same way a bid for a new house might be compared to the established track record of houses previously built by that contractor.

In the case of the DeMarco and Lister quote ("I am the boss, for God's sake! It will be done in two years!"), the negotiation of the date takes on a more substantive dimension when using the objective criteria of a trendline as a basis for what is fair and realistic.

This use of benchmark data as objective metrics criteria is critical to negotiating terms for IT projects. It is designed to create efficiency in negotiations by not wasting precious time on defending one's position and attacking the other side. It is also designed to protect the relationship of the principles, since positional arguing usually involves attack and counterattack. As Fisher, Patton, and Ury state:

> If relying on objective standards applies so clearly to a negotiation between a house owner and a contractor, why not to business deals, collective bargaining, legal settlements, and international negotiations? Why not insist that a negotiated price, for example, be based on some standard such as market value, replacement cost, depreciated book value, or competitive prices, instead of what the seller demands? In short, the approach is to commit yourself to reaching a solution based on principle, not pressure.

This is the role of benchmark assessments for IT organizations. If an organization establishes its own performance baseline, it can plot its own trends, as shown in Figure 4. These can be plotted for that all important metric: speed. Legitimacy of future project deadlines could be assessed. And the problem could be worked backward in the cases where IT organizations have a deadline but don't have all the requirements to determine project size. In this case, one would draw a line at the time of, say, eight months and then read off the best case/worst case scenario for what could be built (project size) within those eight months. It at least brings in a range of values to start the discussion, with the negotiation centering on the

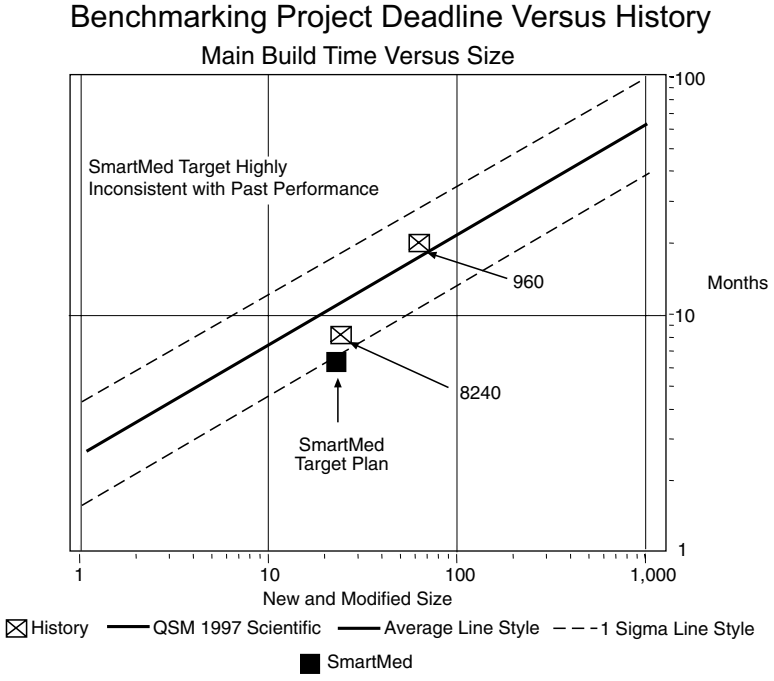

Figure 4 — Using a benchmark trend to assess three projects.

agreed-upon scope for the project. Similar charts frame the negotiation for effort, budget, and staff.

Industry Benchmarks for Process Improvement and Outsourcing

Plotting an IT organization's capability through a benchmark assessment establishes its baseline. This reflects the state of its practice during the time period from which those projects were gathered (e.g., January-December 2000).

If an organization were to maintain a metrics database of its 2001 projects, these could be compared against its 2000 baseline for speed, cost, and quality. If the bulk of these projects positioned faster than last year's performance, say, in the upper quartile, then good things are happening. Similar charts should be read for effort and defects. If these additional dimensions are also improving, then good things are happening across the board. Sometimes this is the case; other times, speed might be improving while costs and/or defect stay the same.

This is the basis for IT metrics forming a balanced scorecard. Whether you are negotiating with an outsourcing supplier for year-over-year improvements or monitoring the payback for your own internal process improvement program, these metrics form the basis of merit and legitimacy for your negotiation positions and improvement targets moving forward.

Summary

Ever-present pressures on time to market call for skill in negotiation. These pressures will be compounded with the belt-tightening that comes from an economic slowdown. IT professionals on all sides will need to enhance their negotiation skills to focus on solving the joint problem they share with the business side (or with their outsourcing partner).

Metrics are critical in this environment. Numbers and charts bring speed, efficiency, and clarity to achieve wise and sustainable deals, whether parties are negotiating a deadline, the project scope within that deadline, or productivity improvement targets.

With numbers, we can focus on the problem at hand, rather than allowing negotiations to become personality-driven, with entrenched positions based on ego and partisan perceptions. Metrics frame the problem, enabling parties to make meaningful and productive comparisons and to shift discussions to underlying interests. Metrics models and databases help teams examine all possible options, and benchmarks help establish standards of fairness and legitimacy.

The organization that succeeds in blending these two emerging and powerful disciplines will find its energies channeled in efficient ways that are not possible in organizations embroiled in infighting and conflict. These organizations will be achieving their goals in the marketplace, while competitors that fail at combining these disciplines are plagued by protracted disputes and failed projects. Which type of IT organization we'll have will be determined by the choices we make in this area.

References

Fisher, Roger, Bruce Patton, and William Ury. *Getting to Yes: Negotiating Agreement Without Giving In.* Penguin, 1991.

Fisher, Roger, and Alan Sharp. *Getting It Done: How to Lead When You're Not in Charge.* Harperbusiness, 1999.

Heen, Sheila, Doug Stone, and Bruce Patton. *Difficult Conversations: How to Discuss What Matters Most.* Penguin, 2000.

Mah, Michael C. "Case Study: Benchmarking, Estimation, and Applications Outsourcing Gone Awry." ***IT Metrics Strategies,*** Cutter Information Corp., October 2000.

Mah, Michael C. "Internet-Speed Deadline Management: Negotiating the Three-Headed Dragon." ***IT Metrics Strategies,*** Cutter Information Corp., May 2000.

Mnookin, Robert, and Lawrence Susskind. *Negotiating on Behalf of Others: Advice to Lawyers, Business Executives, Sports Agents, Diplomats, Politicians, and Everybody Else*. Corwin, 1999.

Mnookin, Robert, Scott Peppet, and Andrew Tulumello. *Beyond Winning: Negotiating to Create Value in Deals and Disputes.* Harvard University Press, 2000.

Climbing the SEI CMM Makes a Difference on Software Projects

Continued from page 1.

process maturity levels to completed projects that had not had SEI-style process assessments.

Mapping SEI Levels to Completed Projects

The first step in mapping completed project information to SEI process maturity levels was to find something in the QSM database that had a distribution similar to SEI's process maturity distribution by project. Figure 5 shows the SEI distribution for projects by maturity level; Figure 6 shows QSM's distribution of its process productivity index (PI). PI is a calculated efficiency metric derived from the size of a project, combined with both the expended effort and the time spent by a team. It differs from traditional productivity metrics that only express output size (like function points or code) over unit effort (such as person-months), while omitting elapsed time.

Note how quickly the SEI maturity level falls off and how quickly the QSM PI falls off after a value of 14. Are these two distributions the same, just with different granularity, or are the differences due to chance?

To test the hypothesis of similarity, we needed to convert the QSM PI to something that looked more like the SEI distribution by combining several PI values into a single maturity level. There are four ways to do this:

1. Apply the same distribution of SEI projects by percentage to the QSM projects.
2. Smooth the SEI percentages and apply them to the QSM projects.
3. Use industry experience to combine the QSM PIs into SEI levels.
4. Use a Bayesian technique to assign QSM projects to SEI levels based on odds.

We selected the fourth method (note that QSM has applied the third method with results similar to the ones reported here). The results are shown in Figure 7. Note that the SEI distribution becomes smoothed using this method of mapping.

There is still a question of face validity. Even if two distributions can be made to match one another, is there any underlying validity in asserting that one is like the other — i.e., is there causal reason to relate the two? We looked at the definitions of software process maturity and PI to see if there was an underlying relationship.

Software process maturity is the extent to which a specific process is explicitly defined, managed, measured, controlled, and effective. Maturity implies a potential for growth in capability and indicates both the richness of an organization's software process and the consistency with which it is applied in projects throughout the organization. The software process is well understood throughout a mature organization, usually through documentation and training, and the process is continually being monitored and improved by its users. The capability of a mature software process is known. Software process maturity implies that the productivity and quality resulting from an organization's software process can be improved over time through consistent gains in the discipline achieved by using its software process. [1]

PI constitutes a macro measure of the total development environment. Low values are generally associated with primitive environments, poor tools, unskilled and untrained people, weak leadership, and ineffectual methods. High values are usually associated with good environments, skilled and experienced people, excellent leadership, effective tools, and sound methods. Projects with high values demonstrate a combination of improved effort and/or faster speed, and lower defects.

These definitions indicate that each of these measures is multidimensional, running through all of the aspects of managing and developing software (e.g., environment, tools, methods, and leadership). Accordingly, we accept the informal hypothesis that both software process maturity and PI are gross, macro measures, possibly of the same thing. Therefore, the mapping of SEI levels to PI figures is logical.

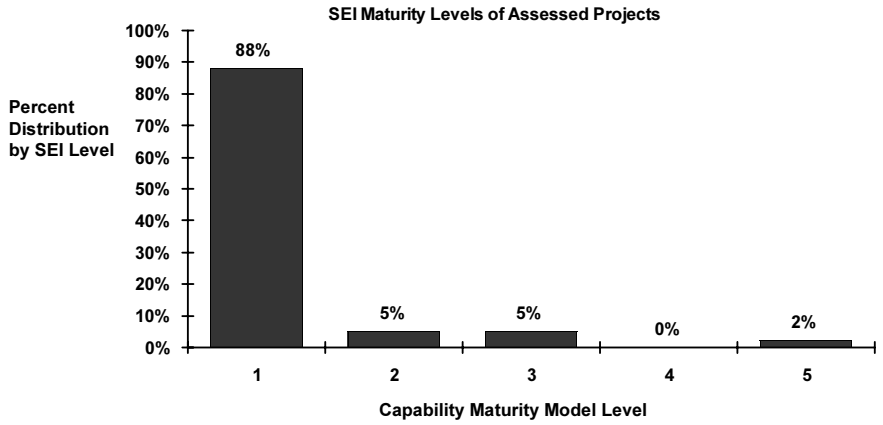

Figure 5 — Software Engineering Institute (SEI) process maturity levels by project for 296 projects.

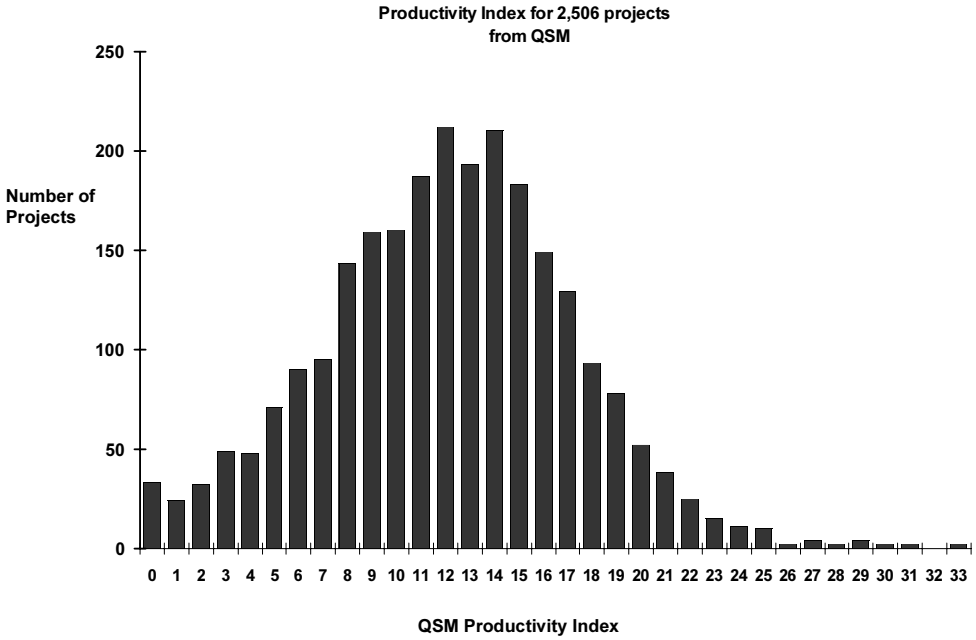

Figure 6 — Productivity index for completed projects.

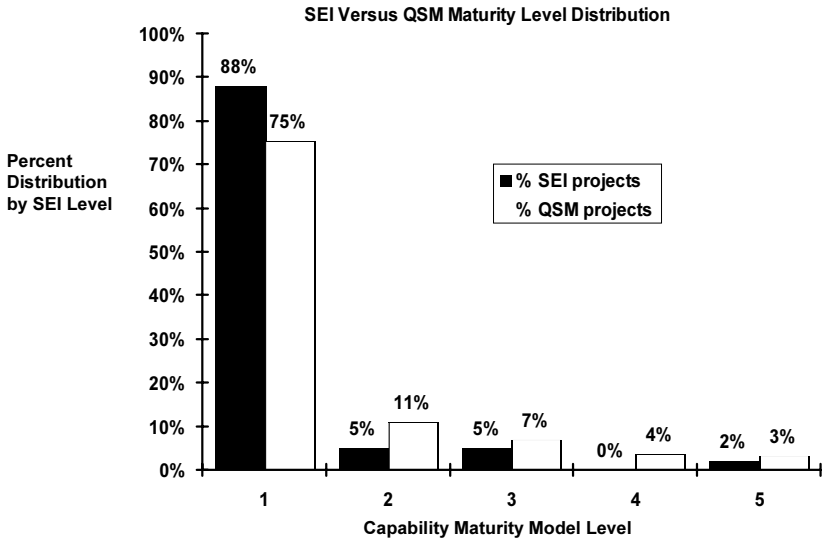

Figure 7 — QSM productivity index mapped to SEI process maturity level.

Results

Table 1 shows the results for the same project at different SEI process maturity levels. The project was constructed to be about 200,000 lines of a business application. There was project data in the QSM database on 1,300 business applications. The figures are only for the (single) phase that includes low-level design, coding, and all testing up to final delivery to a customer in a production setting; this phase is commonly called code and test.

Comparing Level 1 performance with Level 2 performance, we see that there is a 10-month reduction in schedule and a 75% reduction in effort (cost). Also, Level 1 organizations have a ratio of highest cost to lowest cost of just under 100:1. What can account for the 4:1 ratio in effort? We scanned the QSM database for other information that could explain the ratio. The results are presented in Table 2.

As we can see in Table 2, the ratio of discovered defects is 4:1 in comparing Level 1 to Level 2 (discovery includes injection, discovery, and removal). Note also that there is a ratio of 5:1 of shipped defects (i.e., those discovered by the user). The defects include all categories, not just serious and critical.

Conclusion

By mapping SEI process maturity levels to some aspect of completed software projects, we can estimate the effects of software process maturity on project results, which, in the case selected here is dramatic.

This expresses the specific effort, defect, and time reductions that would be observed if organizations successfully achieved the levels of performance implied by higher process maturity.

With actual metrics on improvements in these dimensions, organizations can demonstrate the bottom-line payoff to their efforts associated with better processes, whether those processes are initiated inhouse or with the assistance of an outsourcing provider.

These metrics provide a method of legitimizing the results of such process improvement strategies and help establish quantifiable goals that organizations should measure to affirm the achievement of productivity goals.

References

1. Paulk, Mark, Bill Curtis, Mary Beth Chrissis, and Charles V. Weber. *Capability Maturity Model for Software*, version 1.1, 1993. Software Engineering Institute technical report SEI-93-TR-24. Available from www.sei.cmu.edu.

Table 1 — Project Results by SEI Level
(for the code and test phases only of a typical 200,000-line business application)

SEI Level	Calendar Time (Months)	Effort (Person-Months)	Total Cost (in dollars)		
			Median	Lowest	Highest
1	29.8	593.5	5,440	1,786	101,721
2	18.5	143	1,311	962	1,732
3	15.2	79.5	728	518	933
4	12.5	42.8	392	279	502
5	9	16	146	15	271

Table 2 — Project Results by SEI Level: Defect Rates
(defect counts include all phases up to final customer shipment)

SEI Level	Calendar Time (Months)	Effort (Person-Months)	Defects		Total Cost Median (in dollars)
			Discovered	Shipped	
1	29.8	593.5	1,348	61	5,440
2	18.5	143	328	12	1,311
3	15.2	79.5	182	7	728
4	12.5	42.8	97	5	392
5	9	16	37	1	146

Additional Resources
Putnam, Lawrence H., and Ware Myers. *Measures for Excellence: Reliable Software on Time, Within Budget*. Prentice-Hall, 1992.

Putnam, Lawrence H. *Linking the QSM Productivity Index with the SEI Maturity Level*. QSM, 1996.

Software Engineering Institute. " State of the Practice: A Software Process Maturity Perspective." In *Proceedings of the 1991 SEI Affiliates Symposium*.

Build It Faster!

by Lawrence H. Putnam and Ware Myers

"Surviving in the marketplace means 'first to market.'"
— Stephen E. Cross, Director, Software Engineering Institute, Carnegie Mellon University [1]

"It's still hard to build software quickly, reliably, and of high quality.... Software is increasingly more critical, it needs to be delivered *faster*, and it needs to be of higher quality."
— Eric Schurr, Rational Software Corporation, *Toronto Worldwide Symposium*, 1999

You have probably seen statements like these a few times. You may have even made them yourself when you were called on to make a speech. Faster and better — that's the price we pay for living in a competitive society. But we also reap the rewards. If we are a user of software — and we all are nowadays — we get a wide range of desirable products, faster and better.

Enough of that! Back to the anvil: how can we develop software faster? In particular, how can metrics help us accomplish this goal? Let's take a close look at the relationship that describes software development. First, a general statement:

A *product* of *quality* is achieved with *effort* over *time* at a *process productivity*.

Second, let's substitute metrics — terms that we can measure — for the italicized terms:

$$\text{Size (at Reliability)} = \text{Effort}^{a} \times \text{Time}^{b} \times \text{Process Productivity}$$

(We suspect that the software relationship is complicated, so we put a couple of exponents in it.)

Now we have a relationship that is subject to the rules of algebra. That permits us to rearrange the relationship to show what time depends on:

$$\text{Time}^{b} = \text{Size (at Reliability)}/(\text{Process Productivity})(\text{Effort}^{a})$$

After years of investigation of data from completed projects, we established the value of the exponents:

$$(\text{Time}^{4/3}) = \text{Size}/(\text{Process Productivity})(\text{Effort}^{1/3})$$

In this form, we see that schedule time depends on the other three metrics: size, process productivity, and effort. "Faster" means that we want to reduce time. Algebraically, we can do that by reducing size, increasing process productivity, or adding to effort. However, because of those fractional exponents, all three of these effects are nonlinear, especially the effort one.

Let's Look First at Effort

The first recourse of managers, when they suspect the schedule is tight, is to add staff. In terms of the foregoing equation, they beef up the effort term. They fail to call to mind Fred Brooks' long-standing advice:

"When schedule slippage is recognized, the natural (and traditional) response is to add manpower. Like dousing a fire with gasoline, this makes matters worse, much worse. More fire requires more gasoline, and thus

begins a regenerative cycle which ends in disaster." [2]

Now, looking at the equation, we see the formal justification for his famous — but also widely ignored — law:

"Adding manpower to a late software project makes it later."

The reason is that the equation reduces the effort term to its cube root. The cube root of 3 person-years, for instance, is only 1.44 person-years. Suppose we really go gung-ho on the effort and increase it one-third, to 4 person-years. The cube root of 4 person-years is 1.59 person-years; that is only 10% greater than 1.44 person-years. The equation greatly reduces the influence of effort on time.

At the same time, the equation raises the time term to a small power: four-thirds. That adds to the effect. Altogether, there is a power of 4 ratio between time and effort — our fourth-power law. To put the effect in numerical terms, tripling the effort reduces the time schedule by less than one-third. Among those who have examined this relationship, there is disagreement as to just what this ratio is. Some think it is less than four, but it is certainly large.

Moreover, for a given size project and a given process productivity, there is a minimum development time. That means that no matter how much you increase effort, you can't reduce time below that minimum. What that minimum is, for any given project, depends on the size of the project and the process productivity at which the work is pursued.

Above the minimum development time, the maximum practical schedule is also limited. To put it another way, you can reduce effort to about one-third of that required at the minimum development time by extending your schedule to about 130% of minimum. We reach the conclusion that merely increasing effort is not going to get software delivered much faster.

Next, Let's Look at Size

What else could we do? According to the equation, as well as common sense, if we can reduce the size of the project, we reduce the schedule time.

Bells and whistles. Projects resort to the time-honored dropping of features when they run out of time. IT professionals let things that the customer doesn't need much anyway slide into the next release. With an estimating method that determined in advance that they could not build all these features in the time the customer desired, they could trim these bells and whistles earlier — before they started. Knowing that these additional features would probably come on future releases, the project could plan the architecture to accommodate them.

Series of releases. A long time ago, the software industry used to set up projects on a five-year time scale. That usually turned out to be impractical. The underlying technology changed more rapidly than that, for example:

- Tubes to transistors to integrated circuits
- Mainframes to time sharing to minicomputers to personal computers to workstations to Internet-based devices connected to server farms
- Assembly language to third-generation languages to fourth-generation languages to GUI environments and code generators
- Structured analysis and design to object-oriented analysis and design

Besides the technology changes, the society in which the software was to operate changed. For example, business reengineering came along. By the time the software was completed in five years, it was a poor match to its application.

The answer was to divide a big project into a number of releases. We try to hold each release to no more than, say, two years. In today's economy, the time increment may be much less. In that period, although change does take place, it is usually within the scope for which we can plan. At the same time, we do not bite off, especially in the first release, more than we can chew in the schedule time available to us.

Reuse. The third way to reduce the size of the system, at least the effective size (in new and modified lines of code) from the developers' standpoint, is to not design and implement every part of the system. In other

words, to insert already designed and built components.

Of course, you are going to counter that reuse is a figment of our fevered imagination. And we are going to freely admit that it is not easy. But we also know that more than a few companies are having various degrees of success with what we now call component-based development. You architect your system into subsystems, sub-subsystems, and ultimately components with clearly defined interfaces so that you can insert available components into your architecture.

Time. You are not going to learn how to reduce size overnight by using series of releases or component-based development. Experience indicates that these things take considerable thought over a period of years. If we pursue something of this sort, we might expect to reduce project size gradually. That reduces the time to market.

For one thing, dividing a long time-scale project into a number of releases implies that you cooked up a long-term architecture early in the first release, that is, an architecture on the basis of which you could grow successive releases. If your first architecture, on the contrary, turned out to be impractical when incorporated as the first release, then you might have to start all over again with the second release. Instead of putting out a second generation in two years, it might then take four years. The point is: thinking through an enduring architecture, an architecture that can support the changes that several releases will bring, is not easy. When you can do it, however, you can reduce size and get to market faster.

Finally, Process Productivity

If we can increase process productivity, we reduce development time. That is what you suspected all along. That is what the Software Engineering Institute's Capability Maturity Model is trying to do. Experience indicates that increasing productivity takes time at best. At worst, it is fraught with difficulties. Not everyone has succeeded in doing it.

Still, the companies reporting to our database have been improving their process productivity during the 1980s and 1990s. In business systems applications, the average rate of improvement in process productivity has been 10% per year; in engineering systems, 8%; in real-time systems, 6%. These rates are well in excess of the average productivity gain in the US economy during those years, which is around 1%. Of course, the companies reporting to our database are probably well above the average of all companies. Still, the figures show what good companies can accomplish.

One company in the business-systems database sustained a 16% per year improvement rate for about 15 years. That is the best record we have. It shows what is possible. That is close to double the average.

Another way of looking at what is possible is to view the process productivity gap between companies at the 84th percentile (one standard deviation above the mean) and those at the 16th percentile (one standard deviation below the mean). This middle range excludes the exceptionally good software organizations and the very poor ones. It is the range where most of us fall.

In business systems, that gap is 1,060%; in engineering systems, it is 590%; in real-time systems, it is 550%. Faced with differences of those magnitudes, annual gains in the range of 5%-10% look pretty small. Much more is certainly possible for individual companies.

So, you have process productivity, size, and effort to play with to "build it faster." Just don't expect the thought alone to work a miracle. You have to reason it out and work to make it happen.

References

1. Cross, Stephen E. "A Message from the Director," *Bridge*, No. 2, 1997.
2. Brooks, Frederick P. Jr. *The Mythical Man-Month: Essays on Software Engineering*. Addison-Wesley, 1995.