Are Process Improvement Frameworks Worth the Dysfunction?

Paulk: Proposition. Software process improvement frameworks, such as the Software CMM, CMMI for Development, ISO/IEC 15504 (330xx), and even ISO 9001, cause more harm by the dysfunctional behavior they drive than good by the improvement actions they inspire.

Context. We all know that organizations wishing to be certified against some standard or model may try to build a facade of capability rather than investing in true process improvement. Policies such as “you have to be level 3 to bid on this contract” or “you have to be ISO certified before we’ll do business with you” have driven a significant amount of dysfunctional behavior over the years. This has been a fertile source of ideas for cartoonists such as Scott Adams, as observed in Dilbert. At the same time, we all know that many organizations have benefited from using these frameworks to organize and drive their process improvement efforts.

Rob Austin made the observation in Measuring and Managing Performance in Organisations (1996) that using measurement to motivate behavior will drive dysfunctional behavior unless our measurement system comprehensively
addresses all of the factors that are important. This leads to a philosophical divide between those measurement experts who believe that a comprehensive measurement system is infeasible and those who advocate that we should always work to measure everything that is important.

Given the reality of how people and organizations respond to incentives, it is worth thinking about whether our attempts to characterize the best practices of software engineering and encourage organizations to adopt them are doing more bad than good. And if they are overall beneficial, are they significantly more helpful than harmful? As a co-author of several process improvement frameworks, I have always been proud of the work I have done in contributing to these models and standards, while recognizing that some organizations have done foolish things in pursuit of a “piece of paper” to hang on the wall.

This discussion will not resolve the question, but it may shed some light on where we stand.

**Aree:** I would like to argue that the issue is not the measurement framework, but the inadequate handling of detected anomalies, and nonconforming practices and policies. Why is this so? It’s difficult to say, but in the majority of organizations that have implemented process improvement and that are systematically audited, the weakest part of the implementation is getting all involved to recognize and accept the presence of an event or practice that leads to an unwanted effect in a process or in a product delivered to customers. In practical terms this means that the corrective action established to handle a deviation was (in a host of organizations) badly implemented or simply not implemented. In this context, the implementation of true process change requires the coherent implementation of genuine “bona fide” actions, which lead to coherent change in processes and behaviors.

**Blaine:** One view of driving change based on process improvement frameworks is that it is “top down.” When doing top down the process improvement (PI) facilitators at an organization sometimes answer the question, “Why are we doing this [name of an improvement initiative]?” with the very unsatisfying answer, “Because the CMMI says so.” This causes rifts within development groups. Developers feel extra work is being imposed on them, without direct benefit.

By contrast, driving change based on business improvement goals, for example, “reduce escapes from system integration” or “reduce rework by X percent by Q3 2013” are concrete and can be presented to developers as a benefit.

Also, some (untrained, uniformed, neophyte) PI groups try to create process asset libraries that are one-to-one correspondents with the CMMI (or ISO). This causes enormous grief among developers. These neophyte PI groups fail to understand that the models in the CMMI are not processes. It’s basically impossible to write a procedure for “requirements management,” for example. Mistaking process areas for processes can occur when driving change from PI frameworks.

**Anonymous:** Some companies have achieved ISO 9000 by following the standard literally, without recognizing the true intent of improving quality. ISO 9000 is a manufacturing standard established to control the variability in manufacturing components. However, the manufacturing of software is trivial—it’s called copy file. The creation and development of software is the real challenge. One specific company waved the ISO 9000 flag to impress its clients, and the clients themselves were clueless about what this entailed. Software quality professionals did not participate in producing the procedure documents. The auditor who blessed the procedures was evidently not knowledgeable about software development and best practices.

The company wrote its set of procedures and followed them to the letter. Some of the mandated policies required hindered efficiency and productivity. In addition, many crucial aspects were missing in the software development procedures. Specifically, the company exhibited these dysfunctional behaviors:

- A one-page sheet listing the company quality statement had to be posted at each person’s desk. Failure to post this in a visible spot at each workstation was a violation, and thus compliance was checked by in-house auditors. This provided no value to the quality initiative.
- Engineers could not have hard copies of procedure documents on their desks, because it wasn’t possible to ascertain that this was the latest version despite having a version identifier printed on the cover. The company required
that procedure documents be read online. Many violated this requirement by keeping hard copies hidden in a drawer. This greatly reduced the efficiency of the engineers who referred frequently to the procedures.

- The use of software configuration management was not even mentioned, let alone required. Source code was released to the customers. When the customer had a problem, not only were the developers unable to reproduce the problem, but they were unable to regenerate the executable environment. The company did have a commercial source code tool, but the managers did not give developers time to establish a structure, nor the training on using the tool.
- Not allowed to update the software development procedures (because doing so would make the company lose its ISO 9001 certification), the software quality manager created a meta-doc, a procedure document on how to update existing procedures. It did not get approved.

This company was driven by customer contracts; work was performed only if required by the customer. These contracts specified the needs of the deliverable—not how it was to be produced and definitely did not call for implementing known quality best practices.

**Daughtrey:** Some time ago, near the end of an undergraduate general-education class on critical thinking, I had devised what I thought would be a gripping conclusion to our semester of study.

Unfortunately, I had miscalculated the allocation of credits, and students discovered they could receive an acceptable grade without participating in the culminating activity. To my dismay, approximately half of the students simply quit at that point, leaving the final class-wide event somewhat anticlimactic.

I realized I had seen a clear distinction between extrinsically motivated and intrinsically motivated individuals—between those who sought only the grade and those who valued the learning.

I imagine organizations can also be viewed as falling into either the extrinsic or intrinsic camp as concerns process improvement frameworks. Some will value achieving the ranking, the compliance, the “grade,” while others will seek the improvement and use the framework as a means and not an end unto itself. I’m not sure how an organization acquires one or the other of these motivational orientations.

**Jowers:** Software process improvement frameworks are good entities; they provide management and developers tools with which one may assess a project’s software development lifecycle. That being said, I suspect that beyond an early point implementation is counter-productive. First, I’ll briefly give a few examples of anecdotal evidence for my belief.

Case 1: In the late 1960s, I remember from my first week on a summer job, a project (I was not involved) for an airline being in serious trouble because, according to participants, “too many meetings.” That summer I analyzed payroll software for two major accounts and found that they were poorly programmed (they were written in FORTRAN using floating point and did not sum correctly). I learned three languages and a new machine. All of this work was driven by a need for a timely solution. The following summer, I wrote a complex payroll package to replace the package being used by the remaining customer. After I left for the summer (I did return to work there after graduation from college), only parameters for the package were ever modified.

The “takeaway” from this was that there is a timeliness aspect to completion of a project that is more important than even the quality of it. And that a reasonably trained, motivated software programmer, left to self-manage, can create projects that groups cannot create. Later within my soccer coaching “career,” I learned that recruitment is a key to high performance.

Case 2: In the early 1970s I was part of a team of six programmers. I “managed” us. Our best product put us out of business. In a little over a year, we wrote a complex package that input human blood sample, raw data, matched it with a database of 50,000 pathology reports, and diagnosed that patient; it was wonderful. The company then took out a loan for expansion, but then “the lawyers” convinced our client doctors that they would be sued for using the package.

The takeaway for this was that a small group of highly motivated software programmers, left to self-manage, can create wonderful products.
Case 5: Since selling my company, I have worked as a consultant and as a software developer. During a short stint I went direct with some prior clients as a terminal job to my career. They had sold out to a very large company; this was to be a great career move. Unfortunately, the large company had issues. One division of its company was all about software process. They had written great companywide process manuals that were not used. That division could not get work because they were so expensive.

Oddly, the company’s stated goal was to go “lean.” However, we were too lean for out-of-town management, so the company put a fellow who was working on a project management master’s in charge of the project. Although we had some great successes developing the product, he eventually drove it right in the dirt by micro-managing under a guise of process improvement. Meetings on having meetings, paperwork on paperwork, restrictions on the order that work can be done (so as to match the schedule), all took their toll. When our division was shut down due to the economy and government cutbacks, it was a blessing.

The takeaway was that private agendas of management personnel cannot be overcome. “A little learning is a dangerous thing” (Alexander Pope).

In summary, lean is good in software development. Noninvasive data collection and conversion for management is good. Unless one has studied a process improvement framework and attempted to understand the unintended consequences, one should not go there. For example, I have never had a client who could afford above CMMI 3. However, there is much to be used if one does invest in one. I recommend many best practices, but I always stay lean.

Zubrow: There is a presumption in models such as CMMI that the structure of the processes, in terms of specific and generic practices, and the sets of processes as defined by maturity levels and themes will yield better performance. The question posed is whether these models and frameworks have done more harm than good.

First, harm must be defined. Second, it is necessary to identify who has been harmed. Third, I think it would be useful to discuss how the harm has occurred. I believe the notion of harm is best characterized in terms of performance and opportunity cost.
Ehrhorn: While it is certainly possible that obtaining or maintaining a certification drives one to dysfunctional behavior (so, what about those who put CSQE or CQA or … or even Ph.D. after their name—does this inspire dysfunctional behavior?), I think there might be some other things we need to look at.

As a buyer, I have neither the time nor the expertise to evaluate your software development capabilities. But, I know that if you have a certification, like a CMM level, someone who does have the time and expertise has looked at your work, at your manuals, your processes, and so on, and found them acceptable. So, by putting in my RFP that CMM level 4 is required to bid on my contract, I get, at no cost to me, a supplier who has met at least some level of quality. And, I know the supplier has at least some interest in doing a good job just because they have put forth the money and effort to apply for, receive, and maintain that certification.

I work in a highly regulated environment (nuclear) and I’ve never been allowed to use a printed manual. In actuality, we do go audit our suppliers. But, a typical audit lasts three days with three auditors. The checklist has 19 different areas to look at (one is software). While I might get a decent look at a supplier’s software development methodology, I cannot get nearly as much as someone concentrating just on software development. So, even if some of the requirements to maintain a certification seem a bit silly, I believe that what the certification means and what it can do for a customer—I think they are definitely worthwhile.

Duncan: My initial thought is that you have somewhat answered your own question in saying “some organizations have done foolish things in pursuit of a ‘piece of paper’ to hang on the wall.”

The main thing I see is that all of the frameworks you mention, and many others, had at least some expectation on the part of at least some of their participants/sponsors during their creation that they would be used to drive a program resulting in “a piece of paper on the wall,” at least figuratively. I’ve had direct involvement in the creation of one of the four frameworks you mention and have been involved in the use of the other three to varying degrees. It was clear to me that some form of compliance program was always expected regardless of any official claims made to the contrary. In the case of ISO 9001, there was no question that it was being developed as something that organizations would (at least in some instances) have to meet. Most recently, I note that ISO 15504 has found its way into the TickIT program for ISO 9001 as well.

As you say, of course, it is the behavior pursued by organizations to get the paper, that is the real problem. And that I attribute to the nature of external auditing/assessment since those independently doing that cannot easily know where the real focus should be. They can find nonconformances, but are they really able to identify the most important things that need improving? I doubt it based on my years of experience being audited and doing it internally (where, at least, I know enough to pursue the more important things).

Rakitin: Dave has done a good job of framing what I believe are the most important questions. I recall the issue of Computer Magazine back in September 2001 when the Agile Manifesto was published. When I first read that article, I was flabbergasted. Could competent software engineers really believe that these principles could lead to higher quality software? I certainly didn’t agree and wrote a letter to the editor, which was published and became associated with the “anti-agilists” side of the debate.

The problem with agile and CMMI, in my humble opinion, is that they are both extremes. And after working for 40 years in this industry, one thing I’ve come to believe is that extremes are not helpful. An example—most organizations that claim to be using agile methods are actually only using a subset of the methods prescribed in the methodology. Most agilists agree that organizations need to practice all of the methods to get the most benefit.

If we look at the other extreme, how many organizations implemented CMMI just to be able to bid on DoD contracts vs. out of a desire to improve their development process? The problem with models like agile and CMMI is that they are viewed as the solution rather than a solution. In many cases, what is actually needed is a custom approach that deviates from what the model requires.

The harm that has occurred, in my opinion, is that we have lost the ability to see value in many different
Are Process Improvement Frameworks Worth the Dysfunction?

approaches and then, as a development organization, create an approach that incorporates those aspects of software development that are most appropriate for the kinds of software being developed. Clearly, the process for developing implantable pacemaker software should be much more rigorous than that used for video games.

Jowers: I identified well with Blaine’s comments. And I’ve been where (Anonymous) was. Once, as executive vice president of a hardware/software company, the CEO informed me (in a parking lot) that under no circumstances was I to implement a software quality training program (he was afraid of upsetting the staff). My company had been acquired by this company; this company was a mess with respect to quality and my stated charge was quality. They would deliver software without any testing, had no version control. They even had misspelled words on their screens. They collected lots of metrics but did not analyze them. After my short tenure, their support department grew from five to 25 over seven years; whereas, their installed base had not even doubled. They were so proud of their huge support organization.

Arce: I’m still surprised by the fact that many don’t understand what ISO 9000 is all about. There still prevails a widespread belief that it is not process driven, that its use is restricted to discrete manufacturing processes, and that it requires tons of documentation to implement. Today’s ISO 9000 is lean, process driven, highly dependent on metrics and—more important—worthless without the corrective action and proactive push needed to drive improvement.

McQuaid: Years ago, Tom DeMarco made the infamous statement, “You can’t control what you can’t measure,” and it became an impetus for many metrics programs around the world. But, in a later essay, “Mad About Measurement,” he recanted and wrote “Metrics cost a ton of money. It costs a lot to collect them badly and a lot more to collect them well... sure, measurement costs money, but it does have the potential to help us work more effectively. At its best, the use of software metrics can inform and guide developers, and help organizations to improve. At its worst, it can do actual harm. And there is an entire range between the two extremes, varying all the way from function to dysfunction.” (Why Does Software Cost So Much? 1995)

Measures are still used today for a number of reasons, some successfully, some unsuccessfully. When one is establishing a metrics program, it is critical to identify what it is you are really trying to measure, and how to measure it. But, it is also important to work to identify what unintended consequences can develop from the measurement program.

It is possible to spend a great deal of money on collecting metrics that may not be used, will be used in unproductive ways, or can cause harm to the people working in the organization, as well as to the viability and effectiveness of the organization itself. It is important that people designing the measurement program have a basic understanding of measurement theory. But the organization must not forget about how the results of the metrics initiative will be used. What effect will it have on the people in the organization? Will they become part of the annual personnel review process or used in promotion decisions? Will you end up driving away key people because of a flawed measurement program? Or maybe these people will remain at the organization, but the process may have killed their creativity and motivation. How the results are to be used is a critical part of the metrics process that is often overlooked.

Proper planning and insight about potential problems, and educating management about how to use the measures, can greatly increase the chances of a measurement’s program success.

Our SQP readers most likely have their own examples of measurement programs that went awry. Our participants in this discussion have included some of their own examples, covering a wide variety of industries.
Are Process Improvement Frameworks Worth the Dysfunction?

Dorling: As a result of using the well-known software process improvement (SPI) models, SPI initiatives have been initiated, and experience of implementing SPI has grown. Modest investments in software best practice have produced significant business benefits; however, the focus on model-based improvement has often driven dysfunctional behavior through the improvement actions that they inspire. The capability or maturity levels in the models have become the target without due consideration to selection of an appropriate change management strategy.

The Danish innovation consortium Talent@IT in addressing the problem, identified 10 change management strategies, all anchored in change management literature. For each of the 10 change management strategies, a number of assertions were formulated that reveal for any given organizational setting, the degree to which those conditions are present in an organization. By using a change strategy selection tool the best change strategy can be identified for a given organization and context. A change strategy might vary from socializing in improvements for low maturity organizations, to metrics driven, or by commanding change at a higher maturity.

Recently there has been a generational change in the workforce coupled with the onset of the social media revolution. Generation Y is today the fastest-growing segment of the workforce. Generation Y craves attention in the form of feedback and guidance, and they expect clear goals, trackable progress, shareable status, social visibility, and reward schedules.

The transformational moment then has perhaps arrived where we need to rethink the traditional ways to foster engagement in process improvement. What can one learn from the new generation and the social media revolution? A new approach for a new generation—gamification, a solution for transformational change.

Gamification is the concept of applying game-design thinking through the use of game mechanics to drive game-like player behavior to nongame applications. Gamification is all about injecting fun, recognition, and/or competition into otherwise normal work activities to engage and motivate employees and management to help reach goals.

Game mechanics are the principles, rules, and/or mechanisms that govern a behavior through a system of incentives, feedback, and rewards with a reasonably predictable outcome. Game mechanics motivate people through positive feedback, encouraging engagement and shaping user behavior through the use of points, awarding badges as a social motivator for goal achievement, integrating points and badges into levels creating a sense of forward motion.

It is interesting that in games there are typically 15 to 20 levels, unlike the five to six levels in the process models such as CMM, CMMI, or ISO/IEC 15504 (SPICE).

The underlying psychology of the game mechanics is to provide frequent fast feedback and easy attainment of the initial levels to gain engagement, quick rewards, and achievement, and then to gradually increase difficulty until reaching a target plateau and then make gains easy again, but steeping up at the very high levels. Frequent feedback lets users know how much progress they have made. Progress can be measured at multiple levels.

Using the principles of gamification, a progressive measurement scale can be overlaid on top of the current continuous and staged models. Key value-adding practices can be identified from across available practices in the existing models and allocated to progressive levels of achievement. A system of goal challenges and points awards to attain badges and levels would be set along the path to obtaining traditional process capability and organizational maturity levels, thereby achieving the best of both worlds and retaining linkage to the existing models.

An organization would see progress in terms of points gained, badges received for achieving goals, and attainment of levels on a progressive scale. Even within a level there can be a sense of challenge and accomplishment to have a higher point rating within a level.

Gamification as a solution offers that chance for transformational change with the opportunity for better user engagement, faster feedback of achievement, and more visible progress indicators of process improvement.

I think that using gamification will solve many of these negative thoughts. For the manager and the company, achievement of CMM, CMMI, ISO/IEC 15504 (SPICE), and/or ISO 9001 is a key business driver, a passport to contract awards, a basic minimum license to trade. While the enthusiasm for the “process champions” is initially there, the motivator for the practitioners is not there, and the manager quickly gets into desperation when the levels or certificates
Are Process Improvement Frameworks Worth the Dysfunction?

I believe what has to be kept in mind is that, as Alistair Cockburn has described, the potential of higher levels of risk/loss and increasing sizes of product teams require increased verification rigor and communication formality. However, the agile approach is to start with the minimal set of responsible practices and add to them based on verification and communication needs, rather than start with some very comprehensive set and try to figure out what you can get rid of.

Secondly, I believe, very strongly, that the agile values and principles contain within them very valuable guidance for building a product development approach that seeks very visible, frequent process (and personal) improvement. The details for how to do all of this are not formalized since one of the agile principles is that those directly doing the work should be the ones to consider how best to approach things. (That does not suggest in any way that this be done in some form of ad hoc isolation from any other knowledge, including what frameworks like ISO 9001, CMM(I), ISO 15504, and so on might have to say on the subject.)

I sometimes feel that the strongly prescriptive approaches have buried in them somewhere, at least historically, a belief that people cannot really be trusted to make sensible, proactive decisions and use of knowledge, so they must be told in great detail just what to do and how. On the other hand, these more prescriptive frameworks absolutely contain very useful knowledge based on a lot of people’s experience creating products, improving process, and so on. But I often see the frameworks used, instead of as guidance for people who know how to “drive” as it were, as a way to control people we don’t really trust behind the wheel.

Ferrell: I share your concern on process improvement and possibility of dysfunctional behavior when well-meaning “improvements” are imposed. I think that there are two ways of approaching the topic of improvement:

1. Impose a one-size-fits-all process that strongly coincides with a process improvement framework.
2. Conduct a gap analysis between the existing organizational process and improve strategically and tactically only what needs to be improved.
The first approach is imposed externally and it is sometimes a drastic change to what the teams have cultivated as a “nominal” process. Because it is imposed and because it is drastically different, change is difficult.

My preference is the second type of improvement in which the team is closely involved in both suggesting the improvement and putting improvements to practice. During gap analysis, all can agree on what is deficient and why. Then the change comes from within. In my experience, what the team needs to watch out for are the unfortunate instances where what looks like an improvement in theory has unintended consequences of introducing inefficient workflow, which is prone to introducing errors in the product. I strongly recommend desk exercises to work the process, use checklists and play-act the handoff of products between teams to see if the proposed improvements do not have some inefficiencies. Some tactical changes are easy to make and they yield immediate paybacks, thus encouraging the team. These tactical changes need to support strategic changes. Strategic changes are necessary to make sure the long-term improvements are made and they become part of the culture. However, it is important to take the organization one step at a time along with all of the necessary training to assure that the teams know why and how. In any case, the team must “own” the process and buy into the changes in order to succeed.

Chrissis: Unfortunately, many best-practices models and frameworks require the adopters to use professional judgment and do not come with 100 percent money-back guarantee. They weren’t intended to be compliance models but rather improvement models, which serve as guidelines and best practices for helping organizations to do their day-to-day business activities more effectively and efficiently. However, when they become checklists (aka compliance models), this causes dysfunction in the organization. I believe that most, if not all, standards and models need to be adapted to the organization. That is where professional judgment comes in. If organizations simply use them to get a “score,” most times there is very limited benefit to the organization and any improvement that is realized doesn’t last over time.

Best practices to me are like shoes. You wear different shoes depending on the activity you are doing. And for most of us, we have a few pairs of shoes that we like the best and we wear most of the time. They are ones that suit the purpose, have been broken in to fit our feet, are maintained, and eventually with resistance are discarded when they no longer meet the current needs (for example, tread worn). Best practices like shoes may feel awkward at first. They need to be adapted by people in the organization and continued to be “worn in” over time (aka tailored). Process tailoring is one of the hardest things for organizations to do. It is not like shoes that will “mold” with wear. Oftentimes this takes knowledge and resources, and many organizations are looking for the quick fix, therefore, slipping into the compliance mindset instead of an improvement mindset.

And as we know, when we wear shoes that continue to feel awkward, we usually realize that they are not the right fit and need to be discarded. Many of us are known to have bought a “snug” shoe thinking it will stretch with wear. When it doesn’t, most of us quit wearing that particular pair of shoes and replace them with another more comfortable pair. However, most organizations aren’t willing to discard the processes that don’t work.

One of the points I want to bring up with this analogy is that I have certain shoes that I don’t wear often because I bought them for a particular outfit for a particular occasion. These shoes are usually less comfortable and more difficult to wear. However, this takes time, and many organizations march to an appraisal date because they want that paper on the wall. Most of us can’t get away with wearing only one pair of shoes but need several for different purposes. When organizations adopt best practices because a standard or framework says to do this, and they don’t understand the purpose or have the knowledge and ability to define and tailor them to fit the organization, this is just like a bad pair of shoes. They get pushed to the back of the organizations’ closet and are only used on rare occasions, often with resistance or more commonly not used at all. I make this point because one of the main complaints of these
Are Process Improvement Frameworks Worth the Dysfunction?

 standards and models is that the certificate does not reflect the current practice. Many organizations that only have the compliance mentality quickly abandon their improvement efforts because their efforts were never really improvement, but rather just a paper on the wall. Is this dysfunction or ignorance?

Pryde: I have come across a fundamental question in various forms recently. This question asks whether incentives for certain behaviors will produce more bad or unintended results than good.

Case 1: I work in an Ivy League graduate school. I am on a team that is planning to solicit input from a wide variety of the institutional population regarding enhancements to our existing information portal website. One of the population segments is graduate students. It was suggested that we should advertise gifts and/or food to the students for volunteering to participate in a focus group. Opinions on this question were (to me) surprisingly disparate and strongly held. They seemed to be divided between two contingents:

a. “I’ve been here before and, believe me, students don’t go out of their way unless they see a clear and immediate payoff for it.”

b. “These are adults launching their professional careers. They should be prepared, no enthusiastic, to contribute to a major website initiative. The opinion of people who are only contributing for the reward is likely less than well thought out.”

This is just one more issue that pits pragmatists vs. idealists. View “b” addresses the part of your proposition concerning “dysfunctional behavior they (software process improvement frameworks) drive.”

In the end, it boils down to a cost/benefit analysis of the net value we expect from the behaviors. Do the benefits outweigh the negative behaviors, assuming everyone shares the same desired benefits.

Case 2: Software process improvement frameworks are policies that constrain activities and thought processes to produce certain desirable results or prevent undesirable results. Frameworks are wetware programs. I read an article recently that indicated that such organizational structures can produce the counterintuitive results of shining a light on the more valuable outliers or “positive deviants.”

“Like a species, a workforce can go through a similar evolutionary process driven by individuals with unusual but favorable behaviors.” (See http://www.governing.com/blogs/bfc/col-positive-deviance-government-organizations-problem-solving.html.)

I cringe to think that everybody I work with, newly empowered by this article, should rebel against behavioral norms. “Showing up for meetings on time is now crimping my creativity!”

The point is encapsulated in your phrase “unless our measurement system comprehensively addresses all of the factors that are important.” If we need to look outside of the norms to find the value, then we are clearly measuring the wrong dimensions of behavior.

Paulk: The people invited to participate in this discussion have many years’ experience programming and managing software projects and organizations. Because of the focus on software quality, several have been involved with establishing software quality programs. Some of us have contributed to the creation of various models and standards, including the Capability Maturity Model for Software, CMMI for Development, ISO/IEC 12207, ISO/IEC 15504, and ISO 9001.

As the instigator of this discussion, I find it interesting that some contributors focused on the problems that drove the development of these frameworks. Clearly from their perspective something has to be done to support/drive change. I think all of the participants would agree with that observation, but the question remains whether the motivational use of the frameworks has done more damage than good. I infer that those who focus on the software crisis believe that it has done more good.

Rakitin’s comment that these frameworks can be extremes highlights an important point—that thinking is required to use the frameworks successfully. Those who cannot (or do not) think about how to use the frameworks properly may be capable of rigorously thinking about their processes and products ... but that’s not where the smart money is betting. These frameworks can be useful if intelligently adapted to your business needs.

Which is not to say that informed and competent software professionals may not place their priorities
differently than I would! One of the sad realities of the world is that we all have more things on our plates than we can deal with. Some things are going to fall off, and some are going to be low-priority activities. I’m one of the cadre who argues that a focus on quality and process is a major leverage point for building software effectively and efficiently. As Ehrhorn points out, he does not have the time or resources to thoroughly investigate every potential supplier; certification provides at least a first approximation of capability.

It seems we’ve reiterated arguments made long ago, but it’s worthwhile revisiting these points periodically. To sum up:

- Many software organizations suffer from problems that we have tools, techniques, and methods to address.
- Many software organizations either don’t know about, or don’t have the time to implement, these useful tools (but do have the time to do it over)—thus, there is a gap between the state-of-the-art and the state-of-the-practice.
- Improvement frameworks (certification) can be used to motivate change.
- Sometimes the change will lead to improvement for the organizations that maintain a focus on how the framework can help achieve business objectives.
- Sometimes the change will lead to dysfunctional behavior in organizations that just want a piece of paper.
- Knowing how to build lean and continually improving processes is not a universal skill even for those with good intentions.

**ADDITIONAL REFERENCES**

Dekkers, Carol, and Patricia McQuaid. 2002. The dangers of using software metrics to (mis)manage. *IT Professional* (March/April).

**BIOGRAPHIES**

**John Franklin Arce** is a senior information security systems auditor and is currently the executive director of XPERTIA Chile SpA responsible for the development of proprietary electronic management software tools.

**Taz Daughtrey,** an ASQ Fellow, is senior scientist at the Cyber Security Information Analysis Center, having taught at James Madison University since 2001.

**Scott Duncan** has worked as an agile coach/trainer (Certified Scrum Professional) since 2002, as a quality/process improvement consultant since 1994, and has been involved in all facets of software development since 1972.

**Tom Ehrhorn** has had a 30-year career in software quality assurance with the U.S. Air Force and the Yucca Mountain Project. He is currently a quality engineer at AREVA NP, Inc.

**Leonard Jowers** is a software engineering practitioner and sometimes academician at a local community college. He has developed hundreds of software products throughout a 45-year software career.

**Patricia McQuaid** is a professor at California Polytechnic State University, and is the co-founder and past president of the American Software Testing Qualification Board (ASTQB). She was a contributor to the SPI Manifesto in 2010, and is a Senior member of ASQ.

**Mark Paulk** led the SEI team that wrote the Capability Maturity Model for Software, was co-project editor for the first version of the ISO 15504 reference model, and is a co-author of the eSourcing Capability Model for Service Providers.

**Jason Pryde** is the Web and database manager for the Ash Center for Democratic Governance and Innovation. Prior to joining the Ash team, Pryde held IT staff and consulting positions serving financial, manufacturing, and education.

**Steve Rakitin** is the author of *Software Verification and Validation: A Practitioner’s Guide* and is currently the vice chair for the IEEE Standards working group on Software Quality Assurance (IEEE 730).

**David Zubrow** is manager of the Software Engineering Measurement and Analysis initiative at the Software Engineering Institute. He has researched, consulted on, and taught software process improvement and measurement for more than 20 years.