by Robin F, Goldsmith, JD
robin@gopromanagement.com www.gopromanagement.com

Low-overhead software process improvement techniques can be faster, cheaper, and in some ways better than the well-known high-overhead formal branded initiatives. Organizations can gain considerable advantage from simpler alternatives which actually are implemented and providing improvements, especially from low-overhead methods that overcome often-unrecognized weaknesses in their larger and more complex cousins.

High-overhead approaches characteristically involve expensive, extensive, extra top-down administrative organizational structures to guide the initiative and significant up-front broad-based training of most employees before benefits begin. Ordinarily, the approaches also mandate a number of formal procedures and paperwork be added to regular workflow. Executive support is essential; and conformance often is enforced through command and coercion, frequently resembling evangelical religious fanaticism. For instance, one prominent organization was widely-known for summarily firing not just those managers who privately questioned their branded program, but also those who merely failed to demonstrate adequate active public advocacy for it.

Because of the high start-up costs and relatively long delay until benefits begin accruing, or perhaps because of concerns about the programs' effectiveness once underway, many organizations choose not to undertake these often massive efforts. Moreover, as lead Capability Maturity Model (CMM®[1]) author Mark Paulk said in a November 2002 keynote at the International Conference on Software Process Improvement, "Many—perhaps most—software process improvement efforts fail."

Generic Process Improvement vs. Process Imposition

Many of the big branded approaches include often sizeable and elaborate models of presumably ideal total software processes. Merely becoming familiar with such massive material can be a formidable task, which can pale in comparison to the difficult effort imposing the model processes in one's own organization. Moreover, such overwhelming shotgun-like blanket changes can amount to overkill, possibly replacing existing processes which work fine, perhaps with frankly less appropriate ones, and implementing superfluous processes for things the organization doesn't really need.

On the other hand, generic process improvement analyzes the current processes and focuses rifle-like improvement efforts only on those with opportunities for significant benefit. Ironically, generic process improvement is somewhat of a lost art, largely because the branded approaches have virtually usurped the terms "process" and "software process improvement," though not necessarily with conscious intent. These days, many people think "process" means massive formal procedures and mandated busywork, typically associated with a particular branded model process, and "software process improvement" means imposing that branded model's processes on one's organization.

[1] ® CMM and Capability Maturity Model are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

by Robin F, Goldsmith, JD
robin@gopromanagement.com www.gopromanagement.com

REAL vs. Presumed Process

Although often not explicitly recognized, implicit in generic process improvement is the need to identify and improve the REAL process. *Many (and probably most) generic and branded process improvement efforts fail because they don't properly understand what a process is and therefore don't identify the REAL process that needs improvement.*

Most attendees in my seminars/speeches use expressions like, "How much process do I need?" and define a "process" as "a set of steps one is supposed to follow to turn inputs into a particular result" or words to that effect. That's the definition of a *defined process*, and specifically a defined process mandating formal procedures and paperwork. Rather,

> ***A process is a set of actions, beliefs, and customs that taken together produce a result.***
> A result is the inevitable outcome of the process followed, regardless whether that process was intended, desired, defined, or even recognized.
> Repeating the process produces essentially the same result.

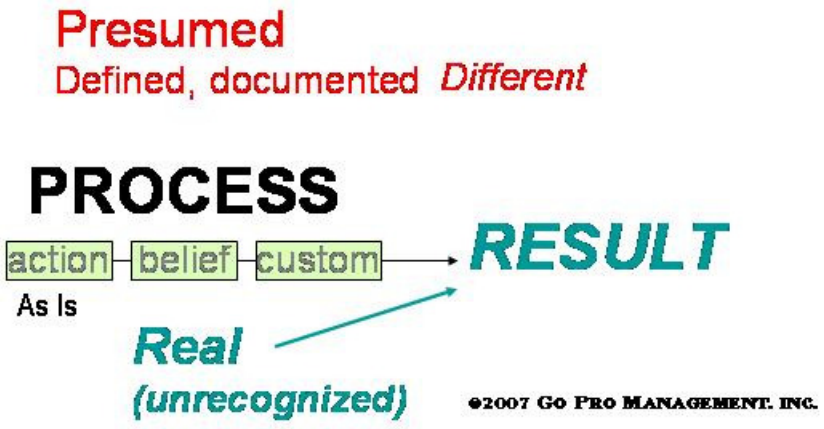

Figure 1. REAL vs. Presumed Process

by Robin F, Goldsmith, JD
robin@gopromanagement.com www.gopromanagement.com

As can be seen in Figure 1, there are usually two processes. The *REAL Process* is the As Is process that actually is producing the results. Very often, people are not aware of the REAL Process. The *Presumed Process* is what people think is producing the results. Often the Presumed Process is defined, which means that different people who are privy to the definition would describe the process in basically the same way. A process can be defined without being documented, although often the process definition is in the form of written documentation; and many defined processes have no formal procedures, or at least none involving paperwork. Customs are examples of defined processes which may not be written down. Again very often, the Presumed Process is different from the REAL Process, which almost always happens when the REAL Process is not recognized.

As Figure 2 shows, when we want to change our results, we must change our process from the As Is process producing our current results to a Should Be process that will produce the desired changed results.

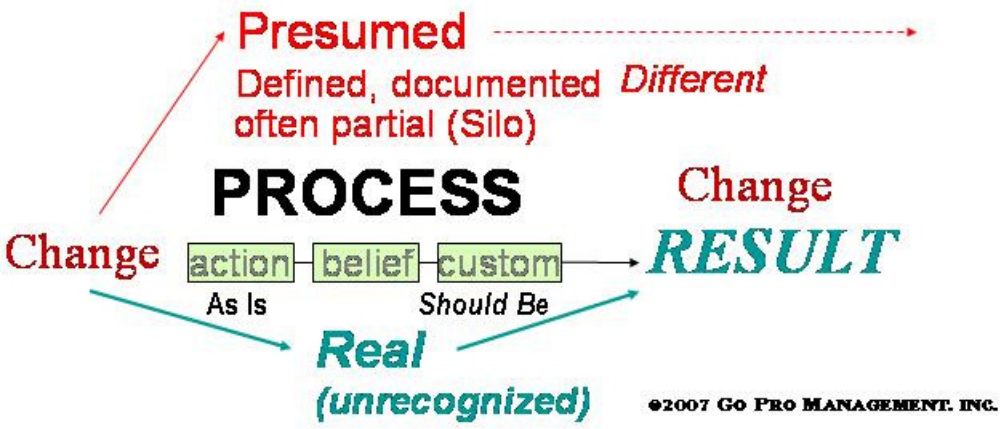

Figure 2. Changing REAL vs. Presumed Process

But, which will people change, the REAL process they probably don't recognize or the Presumed Process they think is producing the results? Obviously, they'll change the Presumed Process; and if the Presumed Process is different from the REAL Process and is not producing the current results, changing the Presumed Process will not have the desired effect on the results. Therefore, to change one's results, one must change the REAL Process producing those results; and before one can change the REAL Process, one must recognize it. That may not be easy.

by Robin F, Goldsmith, JD
robin@gopromanagement.com www.gopromanagement.com

For example, consider the scenario described in Figure 3. I'm sure you are familiar with projects where coding and testing are planned for particular amounts of time which end at the Stop sign deadline; and then the coding takes longer than expected. If the amount of necessary testing is proportional to the amount of coding, and the amount of coding increases, then the necessary amount of testing also should increase. But what doesn't change? Usually the deadline stays the same. Testing gets delayed, squeezed, and cut short of the originally planned amount, not to mention being way short of the revised amount; and the project reliably and predictably delivers trouble.

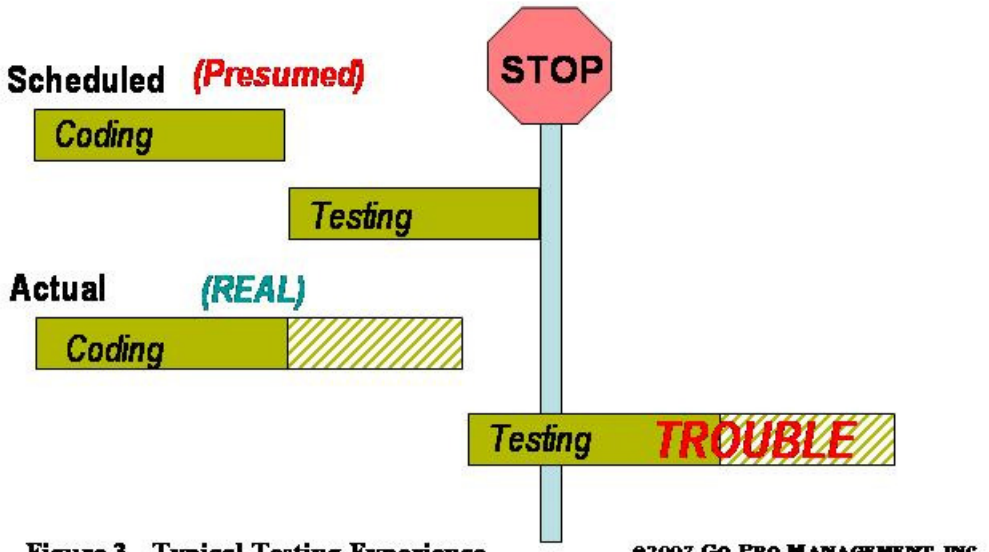

Figure 3. Typical Testing Experience

Have you ever seen this happen? Everyone in IT has. Have you seen it happen more than once? Undoubtedly so. Would it be fair to say it happens practically every project? Most IT folks I meet say, "Yes." Well if this is what happens every project, then your REAL software process is to plan coding and testing, blow your planned schedule, squeeze your testing, and deliver trouble.

Yet clearly nobody in your organization thinks, let alone would say, that's your software process. Instead, they'd say the organization's software process is the Presumed Process: schedule time for coding, followed by time for testing, followed by delivery on the deadline. When they try to improve, they'll make changes to the Presumed Process, which doesn't really happen, and won't be likely to change the REAL Process's predictable production of trouble.

by Robin F, Goldsmith, JD
robin@gopromanagement.com www.gopromanagement.com

Look around your organization with newly opened eyes and you'll see countless examples of unrecognized REAL Processes that differ markedly from Presumed Processes. You should also start understanding why your REAL Process probably includes repeatedly not being able to cost-effectively make appreciable improvements despite making large efforts and possibly enlisting well-known high-overhead branded software process improvement approaches.

Silos

There's a good chance that part of your REAL Process difficulties are attributable to silos. Frequently, the presence of silos makes it especially hard to see the REAL Process. Silos occur when parts of an organization, such as a particular department or function, view only a narrow slice of the REAL Process without being aware of the full process, such as other departments and functions. Viewing a slice of the REAL Process out of context effectively creates a Presumed Process view which is different from the full REAL Process. Thus, to get meaningful context, the REAL Process by definition must be viewed in its entirety, from beginning to full end result.

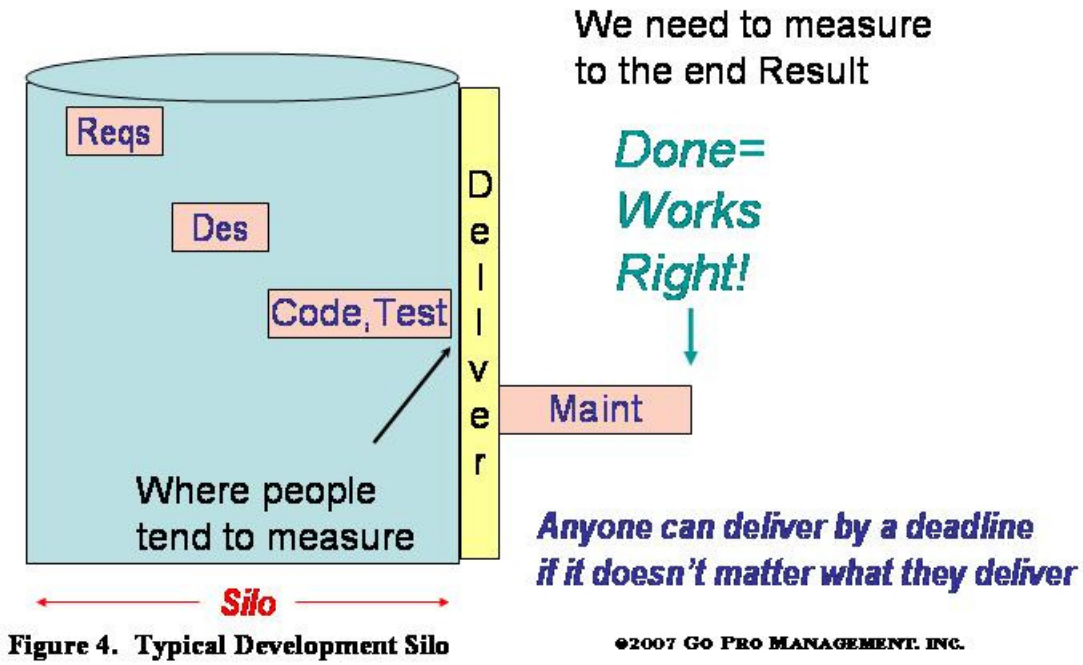

Figure 4. Typical Development Silo

Figure 4 describes the way most IT organizations manage development projects. The project is assumed to end at the point of delivery, and most organizations tend to have a single measurement data point—the deadline.

by Robin F, Goldsmith, JD
robin@gopromanagement.com www.gopromanagement.com

If yours is like most organizations, this is pretty much the way things are done, project after project, which makes it your REAL Process. Dressing it up with lots of paperwork and ceremony doesn't change the underlying REAL Process, or it's predictable problems.

When is the project really done? It's not when the deadline is reached, because anyone can deliver by a deadline if it doesn't matter what they deliver. Rather, the project is really done when the project deliverable *works right*; and everybody in IT knows that almost always additional post-delivery effort is needed to make what was delivered at the deadline work right. You may have a more prosaic euphemism for these finishing touches, but basically it's maintenance.

Your organization also probably accounts separately for the "project" through delivery and post-delivery activities, but they don't tie the fix-its back to development practices that necessitated the extra work. That's a defined procedure which institutionalizes a Process that erroneously Presumes pre- and post-delivery activities are unrelated and thereby virtually assures the REAL Process will stay unrecognized and perpetuated.

Practically every endeavor except IT, and certainly any which is recognized for its effectiveness, knows (1) that meaningful measurement requires more than a single data point and (2) that the consequences of behavior need to be included in the costs of that behavior. For instance, warranty expenses are charged to the manufacturing process. Making the IT delivered product work right is like warranty repairs. The REAL software development process needs to include post-delivery maintenance repairs needed to make the software work right.

If You Don't Know What You're Doing, You Don't Know What You're Doing

You need to know what your full REAL Process activities are, how much you're expending on them, and what you're getting for your time and effort. If you don't have appropriate quantified measures of your results and what's REALLY causing them, you literally and figuratively don't know what you're doing.

A football coach would be fired instantly if he didn't know the score of the game and number of yards gained and lost for each play/formation. Yet, I'd contend ignorance of comparable essential measures is the REAL Process in most IT development organizations. No wonder football coaches get paid in the millions and IT jobs get sent overseas to the lowest bidder.

Be careful of counting on high-overhead imposed processes to cover your bases. Outside of IT, quality and effectiveness are judged by delivered results. For instance, it's widely accepted that a Lexus is a higher-quality car than a Corolla, which is also manufactured by Toyota. It's the cars themselves that warrant the distinctions. The Lexus' superior engineering, materials, and workmanship enable producing the higher quality but are not the higher quality itself.

by Robin F, Goldsmith, JD
robin@gopromanagement.com www.gopromanagement.com

Some other automaker conceivably could be using comparable engineering, materials, and workmanship but still might not be producing a high quality car. Some of the differences might be in the realm of beliefs and customs, especially as promulgated through management practices. Companies that repeatedly produce excellence know their REAL Process includes all these factors.

On the other hand, the quality of one's software is not a criterion for being assessed highly by some of the high-overhead branded approaches. Rather, assessments presume quality will result from following their model process procedures and activities. Moreover, the models and assessments generally do not address beliefs, customs, including management practices.

Furthermore, a seldom-recognized dirty not-so-little secret of most branded big ticket imposed processes is a *giant loophole—management practices generally aren't addressed*. So, after all the worker bees have jumped through all the formal procedure busywork hoops, which may in fact have been helpful producing better software, management continues to make its decisions in the same old same old ways.

The REAL Process of these branded approaches is the presumption that position automatically produces wise decisions commensurate with the preceding process' disciplines. Perhaps you too have heard expressions such as, "Ship it, we'll fix it later. Deadline. Deadline. Deadline."

Two Essentials Your REAL Process Probably Is Missing

We've said a project isn't done until the deliverable works right. The REAL Process in many IT development organizations doesn't include adequate definition of what "works right" means or sufficient suitable determination of whether the deliverable does work right.

In other words, most IT organizations don't adequately know their REAL, business requirements and don't have appropriate Quality Assurance and Testing to give confidence that their deliverables satisfy those REAL, business requirements.

The big branded models don't get into the "how" engineering specifics. That's not their thrust, which is fine, except that they presume dictating that a process should address requirements and quality thereby assures it will be done well. Darn those presumptions.

An alternative low-overhead improvement approach that many organizations use is simply to bypass "software process improvement" initiatives and proceed directly to implementing presumed "good" practices. That's even less delay and overhead than with generic software process improvement. Of course, as with any imposition, there's always the possibility that the imposed "good" practice is not really an improvement and could

by Robin F, Goldsmith, JD
robin@gopromanagement.com www.gopromanagement.com

even be a step backward, which is especially likely if the organization implements and executes the practices poorly.

On the other hand, and emphasizing this big "if" qualification regarding proficiency, implementing some reasonable "good".practices probably will pay off for most organizations. The highest payoff undoubtedly can be obtained by learning to discover requirements, since everything else ultimately depends upon requirements adequacy.

REAL, Business Requirements

Unfortunately much of the IT industry's conventional wisdom is likely to lead astray well-intentioned requirements improvement efforts. The term "requirements" ordinarily is used to refer to the requirements of the product, system, or software that is intended to be created. That's really high-level design of a presumed way *how* to presumably accomplish the presumed REAL, business requirements *whats* that will provide value when satisfied, which seldom have been defined adequately.

For more on how to overcome this common requirements inadequacy, see my Artech House book, *Discovering REAL Business Requirements for Software Project Success*, my February 13, 2007 Requirements Networking Group featured article, "Conventional Requirements Model Flaw Misses Real Business Requirements," at http://www.requirementsnetwork.com/node/683, and my related seminars.

Proactive Testing™

Most organizations also can achieve improvement by implementing more effective quality assurance and testing. While traditional approaches do generally pay off, few people recognize that traditional testing practices tend to be reactive and less beneficial than they could be. Such techniques often come too late and are less effective at identifying test conditions than ordinarily is presumed.

In contrast, my Proactive Testing™ methodology applies a variety of special techniques and approaches that anticipate more of the potential problems so they can be addressed earlier, when they are easier and cheaper to fix.

Rather than being resisted as an obstacle to delivery, Proactive Testing™ wins advocates among managers, developers, and users who find it actually helps them deliver better systems quicker, cheaper, and with less aggravation. More information is available in my seminars and on www.gopromanagement.com.

Summary

Low-overhead generic software process improvement offers attractive benefits compared to high-overhead branded formal process improvement initiatives. The key is identifying and then directly addressing only high-payback issues within the REAL software

by Robin F, Goldsmith, JD
robin@gopromanagement.com www.gopromanagement.com

development process. Short-cut approaches offer likely benefits by bypassing analysis and proficiently implementing presumed "good" practices such as discovering REAL, business requirements and Proactive Testing™.

About the Author

Robin F. Goldsmith, JD

Robin F. Goldsmith has been President of Go Pro Management, Inc. consultancy since 1982. He works directly with and trains professionals in business engineering, requirements analysis, software acquisition, project management, quality and testing. He partners with ProveIT.net in providing ROI Value Modeling™ tools, training, and advisory services. He is author and Course Director for ASPE's two-day seminar, *Managing Real-World Processes and Projects with Metrics*. He also presents a variety of seminars and consulting based on his REAL business requirements and Proactive Testing™ risk-based methodologies for delivering better software quicker *and* cheaper.

Previously he was a developer, systems programmer/DBA/QA, and project leader with the City of Cleveland, leading financial institutions, and a "Big 4" consulting firm.

Author of numerous articles and the recent Artech House book *Discovering REAL Business Requirements for Software Project Success*, and a frequent speaker at leading professional conferences, he was formerly International Vice President of the Association for Systems Management and Executive Editor of the *Journal of Systems Management*. He was Founding Chairman of the New England Center for Organizational Effectiveness. He belongs to the Boston SPIN and served on the SEPG'95 Planning and Program Committees.

Mr. Goldsmith Chaired BOSCON 2000 and 2001, ASQ Boston Section's Annual Quality Conferences, and is a member of the ASQ Software Division Methods Committee and the IEEE Software Test Documentation Std. 829 revision Committee.

He holds the following degrees: Kenyon College, A.B. with Honors in Psychology; Pennsylvania State University, M.S. in Psychology; Suffolk University, J.D.; Boston University, LL.M. in Tax Law. Mr. Goldsmith is a member of the Massachusetts Bar and licensed to practice law in Massachusetts.