There are two battles over process that every small software company must win to be successful.

The first is the battle to convince the company to adopt reasonable development processes. Discussion of what makes up a good process may be an interesting meditation, but is entirely moot until the company commits to a policy of process improvement. The second battle is never over. It is to change existing processes to match changing circumstance.

Within the arena of professional opinion, if not actual practice, the first battle may be all but over. Just a few years ago, the term “process” was used almost exclusively in organizational development programs and human resource departments. Now it stands on the verge of Internet startup buzzwordhood.

But within any particular company the energy barrier between good intentions and actual implementation can be very high. This is often a matter of high expectations that can never be met under typical conditions. Engineers often tend to idealize what it means to “do it right,” a statement of intention usually meant to include such efforts as up-to-date documentation, properly done architecture and design phases, and rigorous quality assurance efforts. When development projects do not manage to produce these things, as most new and small projects do not, the risk is an unintentional abandonment of process altogether.

We suggest a fresh start by considering that the processes by which software is developed are likely to change with circumstance—perhaps even change dramatically—even while general principles like the need for good communication remain constant. For example, the most effective form of code review for a group of three or four developers at the beginning of a product cycle might well be a spontaneous discussion in front of a white board. An appropriate process might be as simple as agreeing on the optimum frequency of such discussions, ensuring that no one’s code is excluded from the discussion, and designating someone to ensure that the discussions actually take place.

We also suggest that the consideration of appropriate process be goal- and risk-driven. A group rushing to develop a prototype in order to obtain angel funding will have a different set of goals and risks from the group engineering version three of an existing product. Understanding goals and risks goes a long way toward suggesting good processes that are, after all, just agreements on how best to make difficult work prone to success.

Note that process improvement for a large company aims to improve efficiency and reliability while reducing costs. Some companies report as much as a 7 to 1 reduction in cost. While startlingly high, these numbers...
are irrelevant to small organizations. A small, growing company would most likely not have the accrued overhead to accomplish such efficiency improvements, and it would be far more interested in controlling the chaos inherent in growth.

Discussion of the best processes to adopt is likely to bring up the choice of pret-à-porte vs. tailor-made. Any number of well-defined processes are available for adoption—Extreme Programming, RUP, and SCRUM, among others. In addition, there are metaprocesses, such as the Capability Maturity Model (CMM), the Personal Software Process (PSP), and the Team Software Process (TSP). Clearly, these processes and metaprocesses operate at different levels with different degrees of formality. Engineering groups have the luxury of selecting one or of developing some variant of an accepted model. The discussion to avoid is the one in which formal processes are compared out of the context of the particular group’s particular circumstances. We believe abstract side-by-side comparisons of process models are essentially meaningless.

An often-told tale has a sitar player asking the Buddha how best to tune his instrument. Leaving aside the question of why a musician would ask musical advice of a monk, the famous answer: “not too tight, and not too loose” is a good general rule.

Recently, a company of about 100 employees that makes business-to-business e-commerce infrastructure software hired a new president and COO. His most visible decision early on was to take the company from its creaky ad hoc processes to a company-wide process for determining what subsequent releases should contain and what the schedule should be, among other decisions. He was careful to emphasize a limited number of fundamentals, such as making sure various departments shared their plans—marketing, professional services, engineering, and so forth—with each other and ensuring that everyone understood what the company was trying to accomplish with upcoming releases.

The actual processes adopted were neither novel nor particularly inventive. Rather they had the virtues of being easy to explain and relatively easy to comply with, with goals easily describable as having been met or not. Subsequent releases under the new regime were successful.

The problem with successful processes are that they, by definition, shift the ground beneath a company’s feet. Success might result in an expanding staff, diversifying product line, new international sales, and more aggressive release schedules—almost anything. Whatever the specific change, the processes that helped create the change in the first place are likely to require overhaul as a result. While this is not a surprising dialectic, it can be difficult to manage and to some extent flies in the face of conventional wisdom that looks to identify a “best” process model. Thus, change management is an essential component of process improvement.

Can a company manage growth-induced change in its fundamental product development processes while simultaneously maintaining enough continuity to keep the process at least minimally predictable and allow employees to plan ahead?

Thinking Objectively
• A process is a tool rather than an end in itself. No process makes personnel interchangeable or substitutes for talent. Nor can a process, by itself, transform an indifferent organization into an effective one. Improvement comes from management assuring that tools are properly applied.

• Processes must be simple. As the previous example illustrated, the new company president implemented a limited number of fundamental processes. Complex processes are difficult to follow, difficult to update, and quickly become unsuitable for the operations they originally specified.

• Processes must be robust. Robustness in this context means a process must be easy to apply and must be difficult to get wrong without warning indications. It also means that as an organization changes, its processes should likewise be amenable to change. Obviously, simpler processes tend to be more robust.

We should emphasize that these guidelines do not argue against adopting best practices, but rather they argue against the prescription of processes and process models without close attention to context.

The claim that the PSP and TSP are models designed for small organizations is an example of a process improvement prescription applied in the wrong context. The PSP is a method for individual improvement, while the TSP, which requires all team members to be PSP-proficient, specifies roles and approaches to development. While these process models show some promising results, they have been implemented for the most part in larger organizations with substantial resources. Given that PSP demands between 100 and 200 hours of initial formal training and an extremely high burden of data collection and analysis, it is unlikely that small companies with limited time and financial resources would risk starting their process improvement program with the PSP.

In summary, processes “done right” for small companies will probably be less than ideal from the viewpoint of both critics and implementers. The exigencies of limited resources and personnel will outweigh the interests of installing full-fledged formal processes at the outset. While the cost of quality may be free in the long run, a small company, especially a startup, cannot ignore its up-front costs. Common sense dictates that processes be established that match the needs of the organization and grow with it rather than being imposed by some abstract ideal.

Mohamed E. Fayad (fayad@cse.unl.edu) is J.D. Edwards professor at University of Nebraska, Lincoln.
Mauri Laitinen (mdl@sierra.net) is a principal in Laitinen Consulting in Lake Tahoe, CA.
Robert P. Ward (robert.p.ward@worldnet.att.net) is a consultant in San Mateo, CA.

© 2000 ACM 0002-0782/00/0400 $5.00