A Reengineering Methodology for Improved Product Development Processes

Dave Brown is a management consultant, teacher, and writer. He teaches management training programs for Support Center University (www.SupportCenterU.com). He also consults with selected clients to establish world-class service operations and is considered an expert in the areas of process improvement, staffing models, and change management. You may reach Dave at his office in Boulder, Colorado, at 303-494-4932 or at dave.brown@SupportCenterU.com.

Have a tough question? Submit your question to Dave at dave.brown@SupportCenterU.com. He will respond to all inquiries, and if your question is selected for publication, you'll receive a complimentary copy of his book, Optimizing Support Center Staffing.

In Dave's May/June column, a reader asked: I'm vice president of product development and support for a mid-size software company. We have total annual revenues of approximately $75 million. My organization includes over 100 people in product support and almost 200 people in product development. When I joined the company three years ago, customers were very unhappy with the product support department. We since have gone through a complete overhaul of our support processes, personnel programs, and tools. We've applied many of the techniques that you've recommended. (I actually attended your reengineering workshop and bought a copy of your book.) We're now providing very good service levels, and we receive mostly good customer satisfaction scores. However, my question is, how can we take it to the next level? Do you have any recommendations on how to integrate support and development processes? Is there any way to apply your reengineering methodology to product development?

See previous issue of *Sbusiness* for the first part of Dave's answer. The second part of the answer involves applying the proven techniques of reengineering customer support processes to product development processes. Clearly, product development is dramatically different from customer support, and there are few processes that would appear to be similar. However, the techniques that are used to identify improvement opportunities are very much the same.

Most of the work involved with being a manager is the same across multiple disciplines. Managing involves coaching and motivating people. It's about setting clear goals and measurable objectives, as well as monitoring performance. You often will hear from first-line people that they expect their manager to be expert in the department's assigned task. "How can someone be my manager if they can't do the job?" Does this sound familiar? Unfortunately, that's the kind of thinking that causes us to promote our best technical people and turn them into mediocre (or worse) managers. But…I digress. The point is that professional managers can apply their expertise and be effective in their roles in virtually any environment.

The same can be said about process improvement. Much of process improvement is based on a set of foundational principles and techniques—

and the methods can be applied across many disciplines. If you understand the concepts behind designing efficient workflow, then you can apply those same concepts to customer service, technical support, or software development (and maybe even manufacturing). While the work that is being performed greatly varies from one function to another, many of the workflow principles are the same.

Although much of the work I've done has been with customer support operations, and most of what I've written about deals with customer support, most of the theory behind my approaches can be applied to a variety of work environments. Let's start by discussing a few key concepts for designing work processes. These concepts also should be the basis for evaluating existing work processes—and they are not exclusive to customer support.

One of the basic premises of designing work processes is the concept of specialization vs. generalization. There are situations that are managed best by generalists, and there are other situations where specialists would be much more effective. If you apply the wrong approach to a situation, you will get suboptimal results. In a support operation, employing only generalists means that anybody can handle any call. That generalization makes staffing and scheduling simpler. It makes call routing simpler. However, it often comes at a price. For instance, you might have 10 different products or modules. An expert or specialist in any one of those product areas may have twice the first-call resolution rate as a generalist. They also may have half of the average handle time as the generalist. That's great performance compared to the generalist model, but it will be more complex to manage.

How would you apply that concept to improve the product development group? One of the most common manifestations of the generalization concept is when product development tries to blend the tasks of developing new products, enhancing existing products, and fixing bugs. Any developer or any group can be working on any one of these tasks at any given time. In fact, they probably would be juggling several assignments at once and, as a result, may be "task switching" on a regular basis. Sure, you can argue that the skills required to find and fix bugs are the same as those required to program new features. You even may argue that the most efficient way of fixing a bug is to give it to the programmer who created it. I won't argue that (although I could). However, I will argue that if a programmer focuses on only one of those tasks, he will become very good at it...measurably better. Most people intuitively recognize that an "expert," which usually means "specialist," can get the job done quicker and better.

So when do you combine tasks and require people to be generalists vs. making the decision to have them specialize? That's the challenge, and there's no simple answer. You need to model the organization in both modes in order to determine which will be most effective. You need to determine how you would organize the work and how many people you would assign to be each type of specialist. You need to consider the fluctuation in daily or weekly workloads. With specialization comes the challenge of load balancing. But don't fall into the common trap of thinking, "The workload fluctuates too much to have specialists, and we therefore need to maintain generalists." That's basically saying, "It would be too hard to manage…so I'll remain inefficient." There are solutions. For example, I've found that many operations can define a core number of specialists for each type of work and then have a smaller team of people that flex between groups based on the workload. In most cases, they would need to cover only two or three skills (semi-specialists, not complete generalists).

The second key concept is focus vs. task switching. You can lose a tremendous amount of productivity as a result of constant interruptions and task switching. Particularly when designing work processes for "intellectual workers," there is a large cost for interruptions to the flow. Anyone who occasionally performs difficult mental tasks knows that focus is essential. You can't perform long, deep-thinking tasks a few minutes at a time; you can't be interrupted constantly, or a two-hour task easily can take four hours—or even all day. This is why we in tech support often separate the phone time from the research/follow-up time. We may go to great lengths, even setting up a separate inbound phone center where the engineers go for part of their day or a separate "Level One" group that takes all of the inbound calls—all in an effort to provide dedicated focus time for engineers to work their open cases.

When you are interrupted, there's more involved than just the time you lose due to the actual interruption. Breaking your concentration creates an additional cost—the cost of getting your focus back…getting your mind back to where you were before the interruption. We call this "re-immersion time." In a software development operation, studies have shown that the typical re-immersion time is 20 minutes. So, a five-minute interruption actually costs you 25 minutes of productivity. In a general-

ist environment, interruptions are rampant. Think about it…one interruption per hour will cost you over 40 percent of your day. Therefore, anything you can do to reduce interruptions and allow your people to focus will improve productivity.

The third concept, task switching, is when someone is juggling multiple projects and trying to work on them in parallel as opposed to starting and completing tasks sequentially. This often comes with a significant overhead cost. One study reported a 40 percent cost (in lost productivity) when switching back and forth between three projects. Please note that task switching is much more likely in a generalist environment (where the person may have to stop working on their project in order to fix a bug).

The real pearl is that many organizations can make a single change and reap multiple benefits. Let's take the example of separating the "bug fix" tasks from the new product development. When you separate these types of work and transform your people from generalists to specialists, each new group will be more efficient in their tasks (because they become experts). You also may reduce task switching and interruptions. Even moderate improvements in each area (e.g., five percent productivity improvement resulting from the specialists, another five percent from reduced interruptions, and another five percent from reduced task switching) obviously result in a significant overall improvement.

Improvements realized in product development will produce a ripple effect that also will benefit customers and those areas that support the customer. This is a complete win for all parties. If you don't have responsibility for product development, here's your chance to collaborate with them for mutual benefit. Start by forwarding this article to the appropriate senior manager in product development. ▼