This article is an excerpt from Chapter 8 of the book "Managing the Unmanageable – Rules, Tools, and Insights for Managing Software People and Teams" by Mickey W. Mantle and Ron Lichty. Chapter 8 - and this excerpt - focuses on establishing a successful programming culture for your team.

An essential and significant element of your role as a great manager is to create and nurture a successful programming culture. For most of us, that's a culture that supports and encourages the delivery of quality software on time and within budget by a team that developers feel proud and gratified to be part of for a long time.

But even if you follow all of our earlier advice in chapters 1-7, it's not easy to manage. Your programmers don't always act rationally or predictably. Some have chaotic personal lives. They don't always get along. They can be blunt, reclusive, irritable, manic, silent, impatient, petulant, abrasive . . .

Your organization may not care much about them (unless their irrational behavior spews beyond your department, of course). But your organization cares a lot about your ability to produce and deploy software that meets organizational goals and customer needs.

Almost any group of programmers, no matter how dysfunctional, will care, too. They care about being productive and building successful products and services.

As for you, you care even more. In addition to wanting what your developers want, and wanting to meet your organization's expectations, you want to be a high-performing software development manager who can stretch beyond the ordinary to achieve the remarkable.

You need help. You somehow need to create internal and external expectations for greatness. You need to instill confidence that you and your team(s) can deliver. You need a culture that supports your goals and objectives. And you need to create an environment of excellence that attracts and retains top talent and motivates stellar work.

Powerful cultures drive high-performance work in ways that no amount of personal motivation alone can achieve.

> *Under the right conditions, the problems of commitment, alignment, motivation, and change largely melt away.*
>
> —*Jim Collins*[1]

OK, so it may not be greatness you need to deliver. For some projects it may be functional but frequent delivery. For others your stakeholders may expect their product to be "flawless." Some teams are formed to help visionaries conceptualize products. Other teams are formed to keep products running as the environments they're built within change. You may find you have organizational goals as well, goals such as developing and retaining quality programmers.

---

1 Jim Collins, *Good to Great* (HarperCollins, 2001), p. 11.

It is essential to creating and nurturing a successful programming culture that you understand what "successful" means for your company, your organization, your project, and your team—and how to measure it.

Unless you lucked out and inherited it, you have to create your own successful programming culture. To maintain it, even if you inherited it, you need to nurture it. These are truisms whether you and your team are developing packaged software, software as a service, embedded software, B2B software components and services, or internal applications for the firm's employees. They are true whether you're part of a tiny start-up, a large corporation, a nonprofit, or government. Your mission is to deliver value. And that requires managing the people and the culture.

Creating a powerful programming culture requires establishing a work setting that is conducive to developing outstanding quality software and values on-time creation and delivery of on-target, customer-focused software:

- An atmosphere of respect and fairness that keeps your staff at their most productive
- An environment in which commitment and motivation are easily nurtured and grown
- Metrics for your products, projects, and deliverables so your team can measure its efforts and improve its results

The challenge is: How do you do that?

All organizations—large and small, companies, governments, and nonprofits alike—have a corporate culture already. It's important to understand your organization's culture in order to create the culture you desire.

If it's a strong, positive culture, it may provide you with a platform you can leverage to create the right environment for your own team. Or it may be one that is so corrosive that you need to wall it off entirely to give your team an insular environment in which it can accomplish good work undistracted.

To figure out the corporate culture, listen to the CEO. But view what you hear and see with some skepticism. What companies espouse is not always how they behave. Look deeper than the words. Enron claimed to stand "on the foundation of its Vision and Values."

Even where values have been forsaken, smart development managers recognize that the company's values, with words painted everywhere, can be leveraged. A good development manager at Enron would have forged a programming culture around three frequently espoused values there, "respect, integrity, and communication," regardless of their absence in the milieu around them.

For organizations less than 30 or 40 years old, our experience is that the culture almost without fail reflects the personal standards and core values of the company's founder(s). Listen to stories from the earliest employees about how the founder established and grew the company.

Ultimately, culture derives not from the words that are espoused, but from the lessons that are communicated through action. Look for how employees, shareholders, and customers are perceived and treated—and the stories employees tell—regardless of the culture your organization claims to live by.

But even in the best of companies, the best of values can be a mixed blessing. Ron's Java initiative at Schwab, at its core, was about building and sharing best practices, about finding and sharing common ways to do things. A slam dunk in an organization with Teamwork as a core value, right? Getting teams to

share best practices, approaches, patterns, and even code should be easy.

But look at Schwab's other values. Striving plus Responsiveness can be translated, in practice, into relentless delivery of customer value. Developers driven to deliver relentlessly will tend to focus single-mindedly on their own work and neither learn from those around them nor, at the end of a project, find time to share.

To make his Java initiative successful Ron had to leverage and emphasize the Teamwork value while deflecting the countermanding pull of the Striving and Responsiveness values. "I found I could get the attention of teams by posing the challenge in terms of results. 'We have a choice: hundreds of one-off projects that take too long and all repeat the same mistakes endlessly; or cultivating a culture of sharing, reaping savings from a pattern of reuse and shared best practices.'"

It should be obvious by now that, even in the best of cultures, you may have to wall off some or all of your company's culture.

You know that achieving long-term success requires that you strengthen your architecture, refactor your code regularly, develop regression tests, improve your processes, upgrade your hardware and software platforms, and do the myriad other things that reduce technical debt and keep you in the game. Mickey often uses the metaphor of oil changes to drive the point home: "Making products is kind of like driving a car. You have to occasionally, but regularly, change the oil."

Yet in some companies, senior management may give short shrift to all but customer-driven projects and ask the question "Why is anyone working on anything that's not an obvious customer feature?"

As a rule of thumb, one MBA can neutralize the efforts of five good engineers.

—*Guy Kawasaki, Early Apple Evangelist and Founder, Garage Technology Ventures*

To be successful in the face of customer-value-only senior management, you need to protect programmers working on under-the-hood efficiencies from organizational disrespect. They need encouragement, nurturing, and praise that will come only from within your organization. They need you to wall off the company's culture, substituting one of your own creations.

Other elements of corporate cultures can be even more toxic to software team success. As one example, you'll need not only to wall off but figure out ways to circumvent cultures that encourage extreme competition and noncooperation among employees.

The final element of organizational culture to understand for your culture-building efforts is: Which function is in the driver seat? Is yours an engineering-driven, a marketing-driven, or a sales-driven company? Is your nonprofit driven by its fundraising or by its mission?

What drives the organization often defines tech's role within it.

Ultimately, the essential balancing act that drives decisions in programming organizations is between being innovative on the one hand and being responsive to customers and markets on the other.

This is a fulcrum of programming culture, but every organization sets it differently. If you get it right for your organization, it will be easier to succeed. If you can't understand where the organization's

fulcrum is set, you clearly don't belong. The fundamental question becomes whether you're trying to serve customer needs or developing technology for technology's sake—technology because it's cool.

Start-ups early on are often technology-driven or vision-driven. But at some point, many if not most companies switch to being customer-focused.

The games and screen savers that Broderbund and Berkeley Systems built demanded lots of innovation, but that innovation was focused almost laser-like on supporting the entertainment value of their games.

Focus on customers is critical for most organizations. When Mickey hears the word framework from a programmer on his team—advocating that the team should build one—his antennae go up. "It's all too often a sign of innovation for innovation's sake," he says.

It is similarly the case that too many programmers are eager to implement from scratch rather than leveraging already written and debugged code. It has oft been said that "programmers are the only people who prefer to stand on the toes of others rather than on their shoulders."

In the end, whether you inherit a positive programming culture or create your own, developing it and keeping it positive requires you to make conscious, intentional decisions over and over.

Celebrate what you want to see more of.

—Tom Peters

Furthermore, it means modeling the culture you're trying to create. Your team looks to you as a model of the behavior you expect and want to see. There may be no stronger illustration of the adages "It's not what you say, it's what you do" and "Actions speak louder than words."

Both of us have known managers who set up programming teams to compete against each other, contending that it is scrapping that motivates programmers to do their best work. We think those managers don't belong in a software development organization. Some of them came from sales. Sales is an individual sport. It's competitive, with salespeople competing against both quotas and each other.

The reward for being a top-performing engineer, on the other hand, is seldom a trip to Hawaii, or golf with senior management. In fact, rewarding competitive performance, whether with those kinds of grand rewards or subtler ones, is almost always inappropriate and counterproductive. Software engineering is a team sport. Smart engineering managers make sure their top performers are happy but reserve the big rewards for entire teams, sending entire teams off on trips and off-sites, for example, as welcome breaks from arduous development marches.

Ron helped turn around a start-up inside Fujitsu that was months late delivering its product. The effort had been based around a culture dominated by a few senior programmers willing to be heroes but expecting to be treated as such. Ron built teamwork out of discord by setting expectations that the work and the rewards would be shared. The team would have to together meet corporate's new aggressive deadline or no one would benefit. "When I arrived, it was a team that had not once met a deadline. But working together, they not only met but beat this one."

In programming, leading teams to their highest levels of performance requires creating cultures that encourage mutual respect, innovation, following standards, expectation of delivery and of excellence, high levels of communication, fairness, empowerment, professionalism, teamwork, passion, customer focus,

and technical excellence.