https://www.infoq.com/articles/create-culture-quality/

Creating a Culture of Quality

Every product company wants to delight customers with a high-quality product, and many engineering organizations naturally focus on process improvements to reach quality goals. But organizational culture eats strategy and process for breakfast. So how do you create a culture of quality?

Product quality is one of the areas that engineering teams monitor closely throughout the software development lifecycle. Most often, a mix of both external quality measures visible by customers– such as defects, reliability, or security – are tracked alongside internal quality measures not visible by customers – such as conformance to a spec, maintainability, efficiency of code, or code base size. When significant or recurring external/internal quality improvements are needed, though, engineering teams often focus on process improvements as the way to get there. After all, the quality of the product is a manifestation of the quality of the process that was used to create it, right?

Products are manifestations of the culture that created them, too. As Bill Aulet, managing director of the Martin Trust Center for MIT Entrepreneurship and a senior lecturer at Sloan, reminds us, culture eats strategy for breakfast – and, I argue, process as well. When organizational culture clashes with the spirit of a process change – such as when a command and control culture tries to achieve productivity gains through self-managed, agile teams – culture wins every time.

Culture manifests itself through an **organization's values, norms, beliefs, and habits** – and these, in turn, impact product quality by guiding team actions. I'm not talking about what an organization says it does; I'm talking about what it actually does - at all levels. To begin with, organizational values typically help prioritize what's most important to teams. For example, an organization that highly values simplicity in design may revert a fix for a defect if it accidentally adds complexity to the user experience. The overall quality of a release may be assessed by the elegance of the implementation of new features it planned to introduce.

Organizational norms define what is considered acceptable behavior, typically through an organization's track record. An example might include whether past releases required extra hours by developers to fix critical defects towards the end of a release. Without having this track record, planning a high-quality release will require a project manager to be over-conservative in estimating team productivity - since a team may not accept putting in the extra hours to meet a quality commitment.

Organizational beliefs are shared ideas or principles about the way things should be. These might include the belief that developers should have ownership and accountability over the code they write – so that their code is reflective of their own diligence and hard work. When major defects are traced to their code, these developers will feel a healthy sense of embarrassment from having missed something and will try to resolve it as quickly as possible. Contrast this with the behavior resulting from an organizational belief that every project and every area of enhanced code will have defects – and these should be expected. With this belief, developers will be slower to react when defects are found in their code.

The last element of culture is organizational habits, and these guide the regular and often unspoken tendencies that teams have. One example is the style with which decision-making occurs. For some organizations, decisions require deferring to the most senior person in the room, while other organizations will define and use evidence-based criteria that people at any level can apply in order for the decision to be accepted. It can be quite instructive to observe how a team decides whether quality is good enough to release – since this is perhaps the most important single decision impacting quality that is made during a release.

So how do you drive quality through culture? In their HBR article on creating a culture of quality, Ashwin Srinivasan and Bryan Kurey of CEB share the results of their recent research on this question. They studied over 850 employees who impact quality from 80 multinationals to find four factors that drive quality as a cultural value:

1. **Leadership emphasis**. Managers need to demonstrate how to "walk the walk" with respect to quality, and this must come right from the top. You can do this by:
   - *Tracking quality metrics*. Define meaningful measures of quality that senior leadership, product management, QA, and engineering can all agree to.
   - *Making your metrics visible*. Pull them out frequently in meetings, and review them in regular check-ins with your team.
   - *Using quality in trade-offs*. Create clear definitions and guidelines for minimum quality levels, and use them in meetings when trade-offs need to be made at the end of a release. When your team sees quality metrics being used in trade-off decision-making, they'll know that quality matters.

A special note is warranted here when introducing or changing metrics in your organization. As with any change initiative, it is critical to balance buy-in with mandates to adopt the change. The risk with metrics is that different teams may begin using their own variations that emphasize their interest over those of others. Since the purpose of metrics is to measure and shift team behavior across the board, it's critical that all stakeholders (senior leadership, product, QA, and engineering) agree and adhere to a common set. You can achieve this by:

- *Forming a cross-functional working group with purpose*. Motivate the need for the metrics by clearly identifying the current pain points that exist without them, why action is imperative now, and how common metrics will help. Invite influential stakeholders from across senior leadership, product, QA, and engineering to design the metrics. Each participant can represent their team's interests during discussions, as well as help sell the metrics internally to others. Choose a good facilitator and be sure to explicitly ask participants to sell the result to their peers after the metrics have been designed.
- *Measuring for outcomes that matter*. Ask the working group to begin by identifying the desired, qualitative product outcomes that different stakeholders expect. Once these outcomes have been identified, invite the group to work backwards to select metrics that measure movement towards or away from each outcome. For example, let's say your product is a cloud application whose computational costs are rising faster than adoption – and senior management is concerned. The working group might identify various metrics to measure efficiency, such as CPU utilization across servers, which can be tracked during development and testing. Once the metrics are finalized and in use, show your teams what impact they are having.
- *Standardizing metric use across teams*. Charge the working group with creating the templates and dashboards that all teams will use to view the metrics. Invite each participant to present the results to their respective teams, and ensure these are used consistently. Since all functions had input into the process, clearly set expectations that these metrics should be used going forward by everyone.

2. **Message credibility**. Successful managers carefully choose the right way to communicate the quality message based on what resonates with their team. This may take some experimentation to get right. Communicate the perspective of different internal or external stakeholders about product quality, and see what lights your team up. Examples might include:
   - *Customer satisfaction*. Interview or survey customers on their overall satisfaction with the product, and include quotes to bring their sentiment to life.
   - *Sales experience during demos*. As any sales rep will tell you, having the product crash during a prospect demo can be very damaging – and awkward for the sales rep. Find out how sales reps experience the product during demos and how reliably they can show it off.
   - *Perceptions of senior leadership*. In many organizations, senior leaders (especially founders) love to play with new product capabilities. Invite them to get involved towards the tail end of a release and interview them about their experience.

3. **Peer involvement**. Your team will internalize quality once they begin engage each other about it – and you can help encourage this by taking various steps:
   - *Create rituals at design time*. During design discussions, help your team develop a routine to evaluate the quality impact of various design options. Provide questions for the team to answer for each design being considered, and after a release show how these contributed to overall quality.
   - *Invite peer assessments*. During periodic status meetings, show your team the latest quality metrics and ask each person to make their own assessment of where you stand. Where is there agreement

and where do conclusions diverge? Regardless of the answer, simply inviting the team to make their own assessment will get them thinking about quality.
- *Encourage pair programming*. If done at regular intervals, particularly between junior and senior developers, this will help encourage discussions around quality at design and implementation time. Encourage your senior developers to discuss this in each pair programming session.

4. **Employee ownership & empowerment**. You can empower your team to make quality decisions and feel greater ownership over the results. To do this, consider the following:
   - *Recognize quality initiatives*. Create individual measures of quality (such as bugs per developer, perhaps scaled by project complexity) and provide visibility and recognition to those on your team who get results. Create a dashboard displayed prominently for each person to see how they stack up against peers. Use this in meetings.
   - *Create competitions*. For a big project, consider awarding prizes to the top performers who implement the highest-quality code. Be sure to announce the competition at the outset and clarify how people will be measured. Have fun with it.
   - *Construct learning opportunities*. Invite team members with the best track record to deliver lunch talks on how they approach quality, design choices they made, and the outcomes of recent projects. In preparing the talk, encourage the team member to show links between their approach to quality during a feature implementation and how customers, sales reps, or senior leaders experienced it.

In my 15 years of engineering and product management experience at small and mid-size software companies, I have found that the 2 most powerful approaches to influencing culture are creating rituals and recognizing achievements. Rituals have the capacity to powerfully align teams behind a common practice (forging a team identity), and rewarding successful initiatives create incentives for team members to feel a sense of ownership over their decisions and focus on what matters most.

My first start up experience brought this home. I was a junior engineer fresh out of graduate school, and our VP of Engineering would define phases of our release cycle to control changes made to our source repository. The "green phase" early on in the release meant that our code base was open for all manner of changes, the "yellow phase" in mid-release meant that only low-risk changes were accepted, and the "red phase" at the end meant that each change had to be peer reviewed to meet a certain quality standard. Looking back, our VP continually broadcast the phase we were in (demonstrating a leadership emphasis), and the rituals of peer reviews in the red phase really helped us internalize a quality-oriented mind-set. At the end of each release, our VP would also invite peer nominations for quality efforts, which wisely reinforced peer involvement while recognizing achievement. The result was a strong sense of team identity and shared values that persist with me to this day.

About the Author

Jonathan Levene is a Boston-based executive coach specializing in leadership development of engineering managers and executives. He is a 14-year veteran of product development organizations in small and mid-size companies and has held prior roles in engineering, product management, sales, marketing, and business development. Jonathan provides engineering managers and executives with tools and evidence-based training and coaching programs on leading effective virtual and self-managed teams, influencing leaders in other functions, and developing technical talent for peak performance. For articles and tools on engineering leadership, visit this or contact him at jonathan@levenecoaching.com.