The Code

Behind Successful Developer Teams

BuildBetter / v2.1.JFM17

The Culture Issue

Max Hobbs
/ Editor in Chief / BuildBetter Magazine

Build/better
v2.1.JFM17

Culture Shmulture

Pop quiz: What do software developers hate more?

a. Marketing
b. Corporate Culture

I don't know the answer, but as the Chief Marketing Officer for a software company, I can't possibly be the most popular guy in the office. Nonetheless, here I am to introduce an entire issue of BuildBetter magazine that's dedicated to the culture of software companies and development teams. If this sounds like pointless management voodoo, bear with me—this is important.

Organizational research has shown that culture is not just a factor of happiness and success at work, it *is* (or is not) happiness and success at work. What's more, culture isn't something that happens to us. It's the reality we construct and perpetuate as we live and interact with other people. It's a critical part of our identities and how we make sense of the world and our place in it.

With BuildBetter, we're passionate about building dev leaders and teams, as well as better performing apps. We were curious if developer teams had unique insights for us on how to harness the perfect cultural "life-force," so we asked several tech companies to tell us about their culture. We heard stories from those in the trenches of building a culture from scratch, those working tirelessly every day to maintain their culture, and those who have traveled the difficult road of turning a challenging culture around.

Where we found exceptional dev team cultures we also found leaders who continuously pay attention to their company's culture, who look for signs of toxicity, and who purposefully inject positive elements into their workplace. Culture is deliberately planned, actively and experimentally executed, and constantly monitored for health and performance—just like software applications.

The following articles will help you and your team better understand and define your current culture and its impact on your business. They'll teach you how to build an effective culture from the ground up and how to spot and troubleshoot problems.

The articles are broken up into three sections:

Section 1: Planning Culture. How to think critically about designing the developer team culture your employees and your business needs.

a1 / The Principles of a Winning Culture

a2 / Inspire Your Dev Team in 5 Simple Ways

a3 / Past the Ping Pong Table: Going Beyond Perks

Section 2: Developing Culture. Stories from those who make critical decisions and trade-offs in the name of great dev culture. Planning is nothing without execution. Developing a great culture has a lot to do with the daily decisions that leaders and teams make.

a4 / The Power of Dev Ownership: 7 Tips to Eradicate Micromanaging

a5 / Continuous Improvement as a Way of Life

a6 / A Box Drawn Too Small

Section 3: Troubleshooting Culture. No matter how talented the individuals on your team, problems can always arise. How do you spot them and how do you make changes? Hear from leaders who've had to work to turn dev culture around.

a7 / Architecting a Dev Team Turnaround

a8 / Is Your Team Designing for a Disaster?

Here's to developing better cultures that help developers develop better products better.

Build on!

Buildbetter
stackify.com/buildbetter

@maxhobbs
/in/maxhobbs

Max Hobbs is currently the Editor in Chief of BuildBetter magazine and CMO of Stackify. As an expert in strategy, design, and communications, he's spent the past 15 years helping build businesses—of every size and in nearly every industry—as an entrepreneur, consultant, and creative director.

[1]

Planning

CULTURE

Craig Ferril
/ COO / Stackify

The Principles of a Winning Culture

Culture. ***Of all the overloaded, over-used, buzzword-bingo, five dollar corporate-jingoistic catchphrases of leadership jargon out there, are there any that are more hyped yet seemingly less tangible than this one?***

"They offer catered lunches, free snacks, a kegerator, and foosball!"

Don't get me wrong—I like those things! But once you're in the door and working at a company, what really matters is feeling like you're part of something special; that you "fit" there.

Whether you're leading a small dev team, are in charge of a large cross-functional crew that spans the entire technology ecosystem, or are a founder who is building everything from scratch, there are real, tangible, practical steps you can take to improve your work environment, without having to clear out a conference room for another foosball table.

Let me start with a few admissions. First, although I have helped craft the cultures at a number of organizations, I'm by no means the foremost authority. Second, Stackify (my current organization) doesn't have it all figured out. We're as imperfect as the next company. To be a part of a company is to embrace a certain level of imperfection. An important corollary to that is that no culture is perfect for all people. Nor should it be. What a culture can be is a vital instrument that propels the success of an organization. Or, on the flipside, propels its undoing.

Culture will happen no matter what. You can be intentional about how you want it to look and help it develop its shape, or you can let it happen organically and become something on its own—for better or worse. I think you're better off taking action to try to shape it to your company and team's benefit.

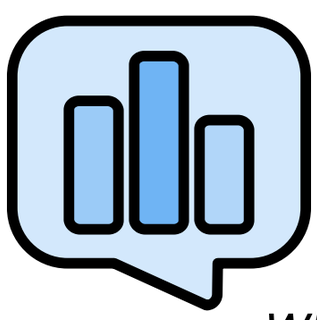
Everyone wants to be on a winning team; to feel they're a part of something meaningful. It's hard to know where to start and how to make strides in practical terms. Here are some key principles that you can apply right away to help you in your journey toward building a successful, meaningful culture. That is, an organization you're proud of, your team is proud of, and your company can grow with.

Principle 1: **Your people create your culture.**

This first principle is perhaps the most fundamental, yet misunderstood, of them all. Your culture is the result of the people interacting within your organization. You can do a lot of things to influence culture, but nothing is more powerful than simply hiring those who match the idea you've created in your mind (or better yet, on paper) of the kinds of people who embody what your "tribe" looks like. Speaking of which…

Principle 2: **Meaningful Culture is Intentional.**

I'm a big fan of Stephen Covey's books, and one in particular that I always go back to is *7 Habits of Highly Effective People*. A core tenet of the book is that we should "begin with the end in mind." I use this lesson nearly everywhere in my life. It's especially applicable when thinking about culture. When you finally get it right, what will it look like? How will it feel from your perspective, and from the perspective of your team? If you can't envision that, you can't really put the first tip to use. How will you know who fits and strengthens your culture if you don't know what's important in the people who are your culture?



What qualities do you value most in a member of your dev team?

Take our poll

With Stackify, I wrote a list of qualities that I look for in the people I want to work around. I knew I wanted the kinds of people who make it fun to come to the office, who I can trust, who want to do great work, and who want to collaborate to build something special. If your people are your culture, your culture is the culmination of their traits and personalities. What kinds of personalities do you want to be surrounded by, and what personalities accelerate your business? This should help guide you with a new level of

clarity when deciding who belongs and who doesn't. And that's important, because...

Principle 3: **Meaningful Culture Requires Tough Choices.**

It's easier to find people who match your ideal than to convert someone who doesn't. You shouldn't cut loose everyone who doesn't exactly match some mold down to the most minute detail, but for the traits that matter most to your culture, you should have a system built into your selection process to identify the existence or absence of those qualities.

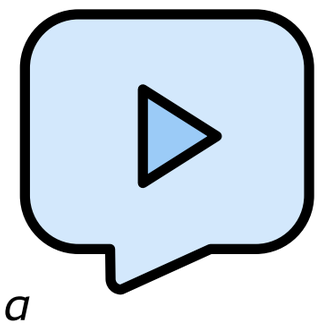
I mentioned that I started with a list of the traits that matter most. The next step was to learn how to spot the presence of those qualities in people who also happened to be otherwise qualified for a given job, and to size up existing people on the team and determine if they possessed those qualities too. You know the people on your team through their everyday work, but it's not so easy in a short-format interview.

The reason you begin with a list of desired traits is partly to guide the crafting of your interview process, and partly to train your eye to more effectively spot the things going on around you already. I decided early on that the single most important trait I wanted throughout any culture I was part of was a strong ownership mentality, mostly because that would ensure a healthy culture of stewardship-style delegation (as opposed to a more dictatorial, "gofer" style of delegation, which frankly sucks for everyone but the dictator). To learn to effectively identify "owners," I had to craft a short list of questions that could be used consistently with anyone I was interviewing, from lead developer to marketing intern to CXO, that would reliably tell me if this person had a track record of owning stuff that mattered and doing it effectively.

Today, I have eight traits that I focus on when interviewing, but still the one I'm most keen on assessing is ownership. After years of asking the same questions of every single candidate, I have developed a keen eye for it. We've baked it into our interviewing process more broadly than just my purview, and we rarely miss. If you were to observe Stackify, you would find a company full of owners. I have never once wondered if someone had something covered that had been successfully handed off to them. Any breakdowns have been entirely on the leadership side, not on the side of our team. Typical, right? It's always the bozos "up top" that are your weakest link. Which reminds me...

Principle 4: **Meaningful Culture Needs a Champion.**

If you've ever planted a garden, you know that the easy part is planting the seeds and the real work is in keeping the weeds at bay. The weeds in your culture happen when you're not nurturing, guiding, supporting, and reinforcing. Just hiring the right people will help you get it mostly right, but don't forget to have conversations with people, to share your perspective on the right actions in a given situation without judgment, to help guide outcomes, and to course-correct when needed. Being consistent about what matters in the day-in, day-out decisions helps reinforce the company's values and ensures that all decisions are consistent with your corporate values. I have yet to see an incredible and resilient culture where there wasn't also real trust and ownership opportunity extended throughout the organization. Help people make the sound decisions you know they're capable of with a little guidance in those moments where things are trending wide of the mark. The rest will take care of itself. Which leads me to the final point…



No one has a devs back like the "spokesmanning" dude in this vid.

Watch video

Principle 5: **Meaningful Culture is Fun.**

Nobody wants to go into an office that sucks the joy out of work. We have an amazing opportunity to contribute meaningfully to the world, to do really cool things with our talents that will benefit others. If you're reading this, you're probably not performing brain surgery or doing something else that is, quite literally, life-and-death work. Your work certainly matters, but nobody is going to give you "that look" if you laugh a little on the job.

Have you ever bonded with people that you couldn't also laugh with? Enjoy the people around you, and let them enjoy the camaraderie that comes with doing important work around a shared cause with like-minded people. If you have hired the right people, given them meaningful responsibilities, and provided them with the right kind of coaching and support along the way, you can trust they will make good choices when you're not looking. When you are looking, well, loosen up a little! You know, come to think of it, maybe that foosball table isn't such a bad idea after all.

Comment:

Stackify

stackify.com

@cferril

/in/craigferril

Craig Ferril is currently the COO of Stackify, a company of developers that make software for developers. He has a passion for applying Lean principles to software and business alike, along with more than 15 years of experience leading software engineering and IT operations for start-ups, SaaS companies, and large enterprises. His background includes leading teams that supported production software.

Adii Pienaar
/ Founder / Conversio

Inspire Great Work in 5 Simple Ways

We make software. We also invest a lot of time and energy in ensuring that the work we do is excellent. It goes without saying then that our development team is a crucial part of our business.

As a team, *how* we work gets just as much attention as the work we do. This mindset recently played an integral role when we rebranded our company to better align with our values, the way we work, and our thoughts about software.

I've always regarded my job as a team leader to be a facilitator of great decision making, not just the one responsible for making good decisions all the time. Since I'm nowhere near technical enough to code anymore, I take a similar approach to enabling our development team to do great work. I help create an environment that inspires success—the rest is up to them.

Over the years, first with WooCommerce/WooThemes and now with Conversio, my approach to working with development teams has evolved. Here are a few pillars that dictate the way my team works and collaborates today.

1. Offer flexible, remote working.
Our team is fully remote, which means that no one can be that pesky manager trying to backseat pair programming or micro-manage projects. All members of the team have the space to tackle challenges in their own way and (mostly) at their own pace.

It's true that diamonds are crafted under immense pressure, but that approach generally does not bode well for software teams. Fixing

the odd bug under pressure is probably okay, but being creative and solving a problem in the most efficient way possible requires time and space.

2. Don't seek perfection.

We're only two years old, and though we've built quite an extensive product in that time, it's by no means a mature product. We've had to accept that neither our development processes nor our code has been perfect.

One of the conscious decisions we've made has been to pursue building features while creating better customer experiences. This doesn't mean that we've taken on a lot of technical debt or had to compromise on scale; we've just prioritized the customer-facing parts of the development process. This has helped us work efficiently while learning fast and fixing our mistakes. This benefit alone seems like a better alternative than having theoretical discussions about coding standards and supposed best practices.

3. Balance costs and speed.

There are many tools and services out there that help young software companies scale up very quickly. The only problem is that the cost doesn't scale indefinitely, which means you're inevitably stuck in that discussion rather than building your own solution. We try not to worry too much about our growing infrastructure-related costs. While a rewrite or refactoring of product parts could probably curtail costs, it would come at the opportunity cost of not working on other things.

Our developers have the trust of the whole team to invest monetary resources such that they are enabled to get the job done in the best way possible. Sometimes that means opting for higher tiers of services in the short-term to give us time to eventually refactor a part of the product. Other times it means delaying a new release or urgently building better scaling into the product to avoid those extra costs. No development team should have to be burdened by monetary decisions, regardless of which route they take.

4. Instill a sense of freedom and mutual respect.

This is one of those things that founders say without offering much explanation. Instead of me tooting our own horn, I'll pass the microphone to one of the developers on our team:

"[Members of] the team all have initiative and are driven to make the product better; we are all invested. I don't know any one of us that would think twice about coming online out of hours to fix something or help a customer, [and] that attitude is what makes our day-to-day better. We have

the freedom to expand our knowledge and experiment, in turn increasing our skills and helping us build better software."

The worst thing that can happen within a team is a culture of apathy, finger-pointing, and "I told you so." We're all human and we all make mistakes. Even the best developers still create software with bugs.

When our performance is about the team and our ultimate goals, it's much easier to respect every individual and give them the opportunity to express themselves through their work.

5. Engage with other departments.

We try to be customer-centric in everything that we do, which means that at various times you can find the whole team helping out with customer support. It's hard to wear multiple hats and switch between doing support and focused programming time, though. We solved this by scheduling on-call developer weeks, where one developer is the primary contact for our customer success team on any technical issues that pop up. The other weeks they can focus on their own code.

It might seem that we have this well-defined process for the way we make software, but the truth is that we have very little in the way of a defined process. Most of the time things probably look pretty messy from an outside perspective. This has been a very purposeful pursuit for us, though. We try to optimize the way we work for every individual, as well as for team and company goals.

I think we're beyond the days when a company and team can expect individuals to adapt to them, but the reverse is never true. We've tried to create a space for everyone on the team to do their best work in the best way they see fit.

Conversio

conversio.com

@adii

/in/adiipienaar

Adii Pienaar knows a thing or two about the inner workings of developer culture. Pienaar is a serial entrepreneur whose career took off with the co-founding of WooThemes and WooCommerce. He's worked on countless projects since then, and made plenty of mistakes along the way. Today, as founder of Conversio, an all-in-one marketing tool for BigCommerce stores, he's taking the lessons he's learned and applying them toward growing a better, healthier, more content team of developers.

Bahman Zakeri
/ Founder, CEO, and Chief Strategist / Xivic, Inc

Past the Ping Pong Table:

Going Beyond Perks

There's quite a bit of press paid to the quirky perks of startup offices. From ping pong tables to beer kegs to pet-friendly work spaces, sometimes it seems like offices are more into having fun than getting work done. But employees who are healthy and appreciated are going to be more inspired, more creative, and more productive.

In order to stay ahead in this competitive industry, startups need team members who have drive, stamina, and, most importantly, a constant stream of creativity. And it takes a lot more to inspire creativity than just a laptop. It takes a culture of support, trust, and, yes, a few extra perks, to bring out the best in employees.

Put people first

At Xivic, we never underestimate the importance of a strong, positive workplace culture, and we see the benefits of our efforts in the variables and outputs from our employees. We consider the maintenance of our company culture to be a task worthy of our full-time attention. It's easy to push culture aside when the projects and deadlines are piling up, but without it, you're not inspiring creativity or accountability, and you're not going to be producing the best work possible. We're constantly asking individuals what kind of output and feedback they're getting and updating our plans based on what they have to say.

We have two offices—one in Los Angeles, and the other in Romania—and we go to great lengths to keep both offices informed and in communication. Our employees are brilliant thought leaders who not only work together, but also genuinely like each other. The communi-

ty is formed with a clear understanding of everyone's roles and how these roles should work together. We're all on the same page. It's important for all employees to be aware of what everyone else is doing, learn about each other's tasks, and think holistically about projects. For companies who are just learning to implement this, I suggest starting small. Try to get people within individual departments to work together on one task, and then start expanding to other departments. It's always easier to collaborate within a department and then grow from there.

We strive to be life coaches for our employees rather than just people who get across immediate needs for work-related projects. To this end, we provide interoffice training every other week. Our last topic was communication—internal, with our clients, and just in our everyday lives. The fun happens after we're in alignment and understand each other's goals.

Take the fun outside

We do a lot of activities, in-office and out. We recently all volunteered through L.A. Works for the Voice for the Animals Foundation and got to hang out with a bunch of rescued kittens. Who doesn't love team-building activities where you're covered in adorable kittens? Before that, my Los Angeles office took a pottery class together (almost everyone has a uniquely shaped vase or jar on their desk now) while my Romania office enjoyed a painting class.

When you get your team doing group activities outside of the office, away from the phones, emails, and deadlines, they can bond on another level. While the term "team building" still connotes a shudder-inducing image of trust falls and name games, giving people the opportunity to bond organically strengthens their work relationships.

Create purposeful perks

You want to support an emotional environment, not just a physical one. The biggest mistake we made in the beginning was not being vocal and discussing the reasons behind the perks we were offering. For example, we started offering free lunch on Fridays, and I noticed that people would come to the kitchen, take their food, and go back to their desk. That wasn't what I expected. We had a meeting and made clear that the purpose was to craft a time for everyone to get together, share a meal, and build our bond. Free snacks and ping pong tables shouldn't be something employees expect, but something they appreciate for what it brings to our work community.

We recently moved our Los Angeles office to a location with more nearby restaurants, parks, and things to do. The goal was to provide a better balance of work and play

for our employees. Sometimes team members will take a break and shoot hoops at the park across the street or, instead of meeting in our conference room, we'll head down to the restaurant next door and soak in the sun.

Trust is the ultimate perk

The team has the flexibility to work remotely when needed, and to come and go for appointments and other life obligations. I'm not a stickler for the 9-to-5 timeframe; I'm more focused on making sure the work gets done. Instill trust in your team and you will get results. We run surveys to get feedback on our culture-building efforts. Even though they're anonymous, people often attach their names. That's a testament to the trusting relationships we've built.

Ultimately, no matter how many amazing perks or free lunches you offer employees, if you don't have a solid foundation and a strong work culture where people feel they are respected, valued, and invested in, everything else is just a temporary distraction. By putting our team's well being and happiness front and center, we create an engaging and dynamic workplace. We expect a lot from our employees, but we set them up to propel the company and themselves forward creatively and professionally.

Comment:

Brian Madsen, of the LIDNUG Talks Team, Time Off and Tools

View post

What office perk is the most motivating for you?

Take our poll

xivic.com

@bahmanzakeri

/in/bahman

Bahman Zakeri is the founder, CEO, and chief strategist of Xivic, Inc. For over 20 years, he has helped create impactful products and measurable marketing solutions for B2B and B2C brands spanning automotive, consumer electronics, healthcare, hospitality, and entertainment. He heads up the L.A. office and lives in the Hollywood Hills with his love and wife, Leila.

[2]

Developing

CULTURE

Jacob Cantele
/ CTO / Concierge Auctions

The Power of Dev Ownership

7 Tips to Eradicate Micromanaging

I can't tell you how many companies I've seen, of all sizes, who start out with an innovative mission to change the world. They acquire a budget, rally a team together, and get to work for months or years with a grand vision. As time goes on, the scope of the project evolves and grows as complex designs are handed to software engineers from on high. Pressure comes down on the software engineers to pick up the pace, and the scope continues to grow. After all, we've waited so long to release this product, what's another few weeks? But the longer we wait, the more funds are expended, and the more teams become stressed.

In teams that claim to be agile, sprint meetings become focused on point velocity and "requirements," rather than actual user stories. Tasks are assigned with very little understanding of the big picture. Someone argues that scrum points should be associated with hours, and someone else says days. In the end, both arguments prove equally worthless as software engineers begin to feel that pointing is just a mechanism for micromanaging how much they accomplished in a week.

Finally the product is released! But virtually no one cares. Customers do not emerge. Frantically, every failure becomes someone else's fault, every success the result of one's own work.

"Let's never do that again," we say. But how do we avoid it?

Concierge Auctions, an online luxury real estate marketplace founded by Laura Brady, was created in order to better facilitate

the auctioning of elite properties around the world. Our purpose has always been to solve problems and make processes easier, faster, and better, and that requires a team culture that operates efficiently and with purpose. Here are the cornerstones of how Concierge Auctions innovated on our own process and built a profitable online real estate marketplace with over $1 billion in sales.

1. Abolish all waterfall processes, especially in agile teams.

If you claim to be agile, design and development must run parallel. The waterfall process assumes that the end product is clearly visible from the beginning and that individual tasks can be neatly checked off and put to the side once they're completed. This misses the point, which is that each step must be done collaboratively, with attention paid to how it impacts other aspects of the larger process. Big design up front is actually waterfall within agile. It shuts down innovation and strategic thinking from software engineers and forces them to spend time on wasteful, arbitrary design decisions. A true agile system leaves room not just for collaboration, but for responding to changes in real time. This in turn saves time, money, and a lot of headaches.

Using Jira and Zephyr to show test coverage

View post

2. Use a Build-Measure-Learn cycle.

Validated learning is the lifeblood of every successful product. It helps us avoid large investments in products that people don't want. Avoid damaging disruptions by building a minimum viable product as quickly as you can that you can put in front of a customer (internal or external) for feedback. This is extremely important for preventing all of the waste that goes into creating products that consumers don't want.

52% of app issues took more than half a day to fix.

View post

3. Death to product backlogs!

No project exists in a vacuum. Product backlogs put too much focus on individual tasks without leaving room for the nuances of a collaborative process. User story mapping is a far superior way to visually map the features necessary for a product to be successful, and to involve customers and stakeholders in creating a successful product vision.

4. Stories, not tasks.

Telling someone to do something may end up with a task getting done, but will it be done in the best way possible? Everyone in our business is urged to think strategically about what we are doing, who we are doing it for, and why we are doing it. During our sprint meetings, we point to the complexity of stories and the business problems we are trying to solve. We do this rather than prescribing a particular solution to be carried out as a task from on high. This

instills a sense of responsibility in our team members, and encourages everyone to work together toward achieving big picture goals.

5. Build cross-functional teams.

Treat people as entrepreneurs, not automatons to be handed tasks. The traditional tech/marketing/sales dichotomy breeds unhealthy company cultures, with individuals across all teams becoming siloed and unable to think strategically about business problems. Every end product is composed of the cumulative work of each team, so why should they function separately?

6. Celebrate every MVP.

Every piece of validated learning, even if it identifies a failure, is necessary in disruptive organizations. Those who are afraid to fail—and to be punished for that failure—will never take the leap of faith necessary to try something new. And what is the incentive to do great work if nobody is even going to acknowledge it? We make sure our employees know they have the freedom to innovate and that their innovations will be recognized.

7. Trust and ownership, not command and control.

Treat people with dignity and respect. Never micro-manage. Give people space to solve problems you didn't know the business had. Simply assigning out tasks isn't enough to build accountability, and you can't expect employees to take pride in the work they produce if you're not giving them room to grow. At Concierge Auctions, we give all of our software engineers 10% of their time to work on experimental projects, which are presented at the end of the month. Successful projects become official projects of the organization.

To build healthy cultures in today's world, we need to enable everyone to think strategically, to be truly agile, and to work in tiny iterations that allow us to learn in a living way. Instead of gauging how much code a person wrote or some other arbitrary metric, let's gauge how much all of us contributed to solving important problems.

Comment:

CONCIERGE AUCTIONS

conciergeauctions.com

/in/jacob-cantele-202286a5

Jacob Cantele is the chief technology officer at Concierge Auctions, a job that goes far beyond just managing the website. Jacob is a systems architect and digital media expert with a strong interest in Agile processes, inbound marketing, and information technology. Aside from his creative talents, Jacob successfully oversees Concierge Auction's team of developers.

Dimitar Karaivanov
/ CEO / Kanbanize

Continuous Improvement as a Way of Life

From the beginning, it became clear that Kanbanize would need more than just cool perks and a nice office to build a dream team equipped to deal with the challenges of a growing startup. A productive and healthy company culture is not something that will happen on its own—you have to build it. Negative culture, though, somehow does evolve on its own, often as a result of disengaged management or a lack of effort toward nurturing a good working environment. Of course, even if you have the best of intentions there will be mistakes and poor decisions. Your job is to not let them drag you down. Act fast and fix what you can.

A quintessential part of the growth and success of Kanbanize was, and still is, our internal culture. Our culture philosophies are value-centered, with attention paid to efficiency and effectiveness. We're all about promoting continuous improvement by nurturing personal responsibility and mutual support within our team. We've learned many hard lessons, and from them we're able to put together what we believe makes our company a great place to work better, build better, and deliver faster.

1. Live what you preach and don't compromise your values.
From day one, Kanbanize followed the principles of Lean methodology and Kanban in all of our departments. Every employee actively applies these lean values and the product we develop in all of their work. We don't just preach and pitch Kanbanize to others, we live it every day.

Based on the combination of the five principles of Lean and the visual nature of a Kanban board with its cards and just-in-time delivery

principles, we've managed to establish a consistent, reliable, and efficient flow of work across the organization. We admit that we are far from perfect, but we believe perfection is in the process, not the destination.

Seeking value, we only put time and effort into things our customers are willing to pay for. Every task and process at Kanbanize either brings direct value or is a value-supporting process (also referred to as necessary waste). Having implemented Lean practicing in our departments we can allow ourselves to spend more time with our clients collecting direct feedback. This is possible thanks to our Customer Support unit and the way it's organized. Every single support request is reviewed by our senior executives in addition to our customer support team. We look for inefficiencies in the product and sort them out as needed, be it through discussions or additional training. In addition, customer requests are copied to the product management queue so management can acquaint themselves with the ticket and prioritize them appropriately.

The Five Lean Principles of a Kanban Samurai

View post

Development processes should be easy on developers and profitable for the whole company. To achieve this, we break down features and tasks until they can't be made smaller without losing the value our clients seek. Projects don't die at Kanbanize, because we only work on strategic tasks and customer requests and we never abandon what we've already worked on before it is delivered to the customer.

These bugs are a lot easier for everyone to see in the light.

Watch video

Keeping the balance between customer demands and strategy isn't easy. The ideas our clients bring are often reasonable and make sense, but implementation can mean taking attention away from strategic features. To make the right decision about the level of priority for each task, we look at the final effect, benefit, and value the feature would bring to the product, our customers, and our company. This means that when a dozen customers really want a certain feature and we believe it will contribute significantly toward the value of the final product we'll prioritize it ahead of a strategic feature that we also consider valuable. As we grow we envision two dedicated teams, one dealing with customer demands and the other solely with strategic features.

Kanban boards add transparency to our work. Displaying every task as a card on a digital board with explicit policies on how tasks should be handled helps everyone see the real-time progress of the team. Multitasking is strongly discouraged—being busy has nothing in common with being productive. The human brain simply works best when doing one thing at a time. In fact, researchers say you can lose as much as 40 percent of your productivity if you mul-

titask. We manage limits on work-in-progress as we would manage any other resource, allowing a limited number of cards for each column and person.

Instead of trying to estimate work items, we track the team's actual throughput and success metrics so we have reliable and predictable measures of productivity. With these reality-based metrics, we always have healthy expectations about the speed of delivery for every kind of feature.

We don't waste time in meetings that drag. The only meetings we encourage are the morning stand up meetings to sync the team's daily goals, regular KPI review meetings, one-on-one feedback meetings, and meetings of strategic importance.

2. The health of your team is the health of your company.

The principles and methodologies we follow are great, but they won't stick unless people respect them. At Kanbanize, the glue holding it all together is a set of specific soft skills and principles that we adopted as the core principles of our team.

As presented in the Responsibility Process of Christopher Avery, when facing a failure, people first deny it, then try to lay blame. Some stop there. Others try to then go on and justify their actions only to later accept their guilt and feel the shame. Out of obligation they will then try to resolve the issue. The constructive and productive work happens when a person steps over their pride, takes responsibility for the mistake, and begins the necessary measures to fix the issue and prevent it from happening again. We encourage people to be objective about their work so that they can take and give constructive criticism.

Every new team member is introduced to the principles and values that form our culture before they get to work. We let them know that criticism within the organization is always shared directly and without emotional attachment, and that feedback is a two-way street. That keeps everyone on the same track and prevents things from getting personal.

Our belief in shared leadership helps facilitate this process. We have regular company-wide meetings where everyone gets updated on the current state and is encourage to voice their views, suggest changes, or share praise or professional criticism. At the team level, we have weekly KPI meetings where we review and evaluate our own work and take appropriate actions to make sure we are on the right course. We also have bi-weekly 1-on-1 meetings between team leaders and individual specialists. These

allow us to share comments and mutual feedback on everything from personal productivity and success metrics to ideas and suggestions for improvement.

Perfection is a journey, not a destination. It's continuous improvement with an eye toward looking for better ways to achieve excellence. The idea is not to settle, and to always look for opportunities for future improvement. It's up to each individual department to meet their own definition of perfection. In RnD, it means code is fully complete and has passed all tests. In the marketing department, success is measured in audience feedback. Across the board, we all do our best to improve each day.

3. Hire extraordinary people.

We learned the hard way that there is no bigger mistake than hiring someone who is extremely good at their job but has no respect for your rules and values. If you want your team to succeed, only hire people who buy into your values, philosophy, and culture.

The people we hire are active thinkers. They're excited about the purpose of the company. They exhibit potential. Taking a cue from the U.S. Navy Seals, we trust our people to lead at ground level and take initiative. We constantly ask for commentary and suggestions on everything from the product itself to the way our organization operates. This bonds the team even further, ensuring every member knows that their role contributes to the prosperity of the team at large.

To make sure our team doesn't lose the drive and passion for what we do, we encourage learning and sharing knowledge. Found a book you believe will further your professional expertise? We'll make sure to order it for our office library. If you read five books from the library in a year and manage to share actionable insights from them in front of the team, you will get a 100% salary bonus at the end of the year. Simple as that. Remember, it's people, not numbers, that make your business as successful as it can be. Creating a stimulating and rewarding environment is in everyone's best interests, so don't shy away from it.

kanbanize

kanbanize.com

@dimitarkaraivan

/in/bdimitar-karaiva-nov-7b32767

Dimitar Karaivanov is the Co-founder and CEO of Kanbanize, where culture is the name of the game. In fact, it's the basis behind what they do. Karaivanov knows the all-around importance of a culture that adheres to its values and facilitates their employees to produce great work, and with an eye to the big picture he's put in dedicated effort to perfecting the methodology.

Armen Margaryan
/ CEO / Margasoft Corp.

A Box Drawn too Small

The Way Developer Teams will Work & Dominate in the Future

As we inject more new technology into our everyday lives, software development continues to be one of the most popular and fastest growing job fields. But have you ever really thought about the core essence of a developer's work?

Software developers tend to think and work only inside the tasks they are assigned to. If they are to write a code on a specific project, they only work on the coding process. As the CEO of a software development company that has been in the industry for more than a decade, I believe that this attitude from developers results only in the loss of career growth. Why do we call them "developers"? Why not "coders" or "programmers"? Because the word develop implies that you are helping something to grow. Developers are supposed to help the whole project move forward and succeed, not just write a code and let it go.

Take a look at this average software development workflow:

```
Requirements -> Design (UX/UI/Graphic) -> Coding -> Testing -> Deployment -> Support
```

Developers must feel responsible for the successful execution of each step of the workflow. The overall success of their code depends on it. When they put themselves solely in the coding "box," the quality of their work becomes invisible. It may look like they did a great job even if as a team they failed to deliver the solution the company needed.

Teaching developers to question

Developing a product means talking to your team members, asking BAs and designers questions to see if they have understood the customer well enough to start coding. You need to talk to QA specialists and deployment and support teams as often as you can to make sure they take on the workflow in the right way.

Learning to question can be pictured with this analogy: Let's say someone asks you to bring them a cup of water. What do you do? An aware developer will ask, "How is this water going to be used?" If you need it for drinking, it will be brought to you clean and fresh at a cool temperature. If you need it to water a plant, it will be brought to you at room temperature and probably not even in a cup. If you want it to make some ice, perhaps there's some ice already made and it can be brought to you directly.

Knowing who, how, when, and why the system is used allows developers to make the code resilient to changes and come up with innovative ideas to make the system more robust and efficient.

Some people may argue that this kind of developer work cycle may be urgent only in small or mid-size companies, but I don't adhere to this opinion. Even in large companies developers need to be aware of the whole work process of the team and the project. Only in this way can the development team succeed and enjoy the good feedback and flow of gratitude from their customers.

Eliminate silos, eliminate errors

There was one project we had where QA teams had automation scripts and developers kept changing the UI elements in a way that continually broke the automation. As a result, QA took longer to find and report issues to the developers, sometimes lagging behind by as much as month. Fixing a bug on a month old code is unpleasant, and bears the overhead of merging it back into the current code base. To solve this, we implemented a naming pattern for UI elements so that the QA team could create automation scripts using that pattern and add logic with parameters. We also started to help QA update their scripts if they were too behind in order to push our code through QA sign-off faster. This eliminated the need for old code bug fixes and nasty merges.

The app lifecycle is everyone's job

Ultimately, developers have to understand and take responsibility for the lifecycle of a developing product. After product deployment, the customer will either curse you or bless you. You: the developer! Not the designers, business analysts, or QA engineers, regardless of whether

the project and collaboration suffered due to those teams. As the person responsible for the overall project, the customer sees only the developer.

As a CEO, I find that reminding this basic fact to Margasoft developers from time to time is one of the keys to the growing success of our company. At Margasoft, developers share the problems of their team members while working together to find solutions. They think as one team. They step up and take responsibility.

People over product

I always like to tell my employees that they develop customers' lives, not just code. We tell them the customers' stories and show them their pain. The number of working hours and lines of codes don't matter to me if the developer doesn't pay attention to the end result: the product, the cornerstone of the change in someone's life. When possible, I invite the developers to user demos so they can hear feedback directly from the customer. It helps with motivation to see and hear the person who will be using the end product. Extra effort is rewarded, both with bonuses and sincere gratitude from end-users.

If you want to be important to your company as a developer, you must put yourself in the shoes of your customers, managers, BAs, and designers before you write your first line of code. Every line of code must be connected with a customer need. If you don't know the essence of the task then ask the right questions. This will enable you to work as a real engineer, a creative thinker, and a truly responsible team member.

Comment:

Getting to Production Faster with More Confidence

View post

the new next

margasoft.com

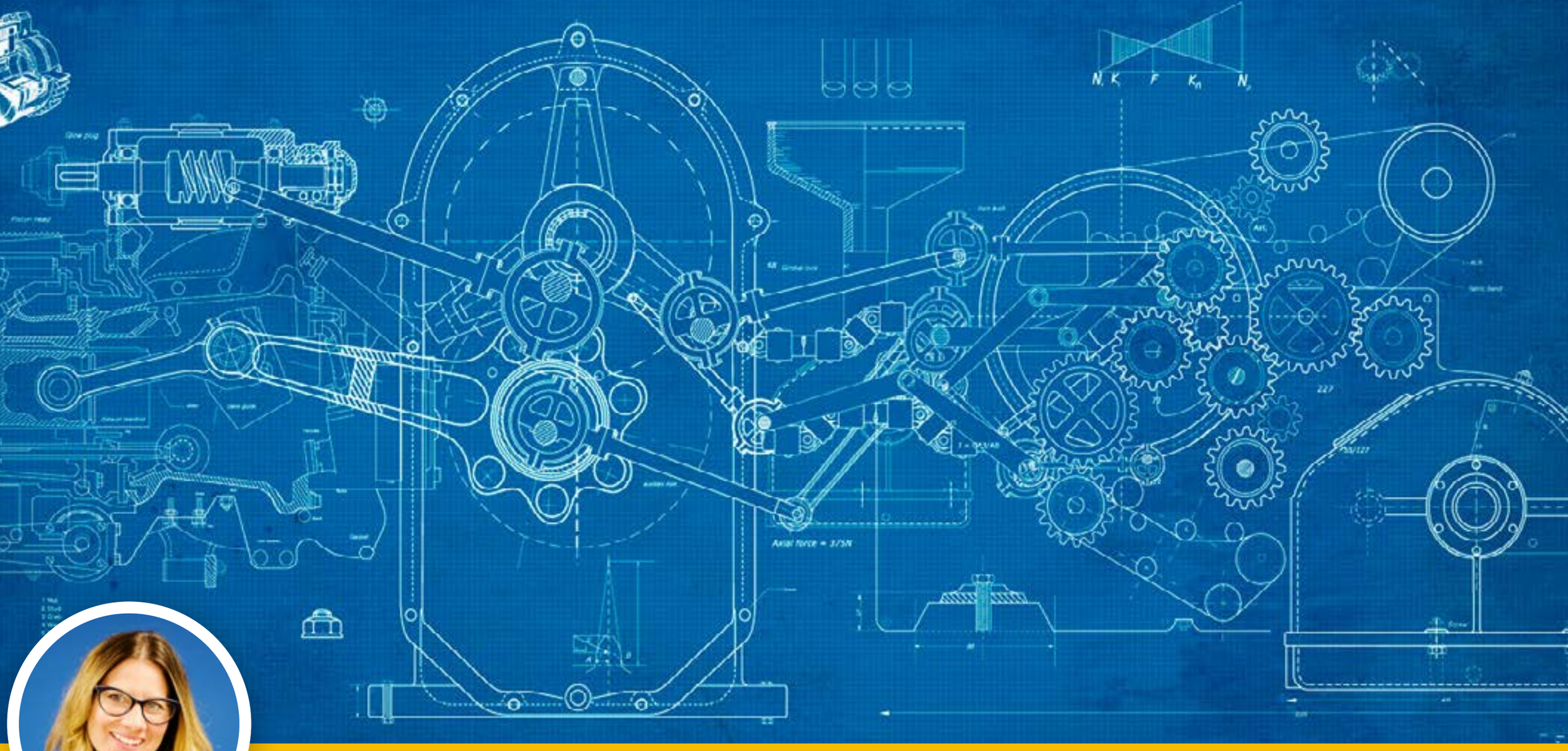
/in/armen-margaryan-8b99093

Armen Margaryan is the CEO of Margasoft Corp, a software development company that has been around for more than a decade. Armen is an engineer at heart who loves setting high goals and enjoying the success of reaching them. He is passionate about working with clever, intelligent, and creative-minded people, and likes seeing the impact of his ideas on clients.

[3]

Troubleshooting

CULTURE



Jayme Thomason
/ Founder & Chief Marketing Scientist / Brink Insights

Architecting a Dev Team Turnaround

A couple years ago, I accepted a position as the "director of digital" at a small marketing firm. And yes, there was still a divide between digital and traditional work there. I was mostly hired for my strategy capabilities and my successes with analyzing digital web properties and finding areas for growth. The title was just that, a title, as I had no direct reports and was only responsible for my own work.

Like many people in leadership positions, I found myself there accidentally. After several months as the resident strategist, I was appointed the leader over the tech team, which consisted of a few developers, a BA, a handful of digital technologists, and the interns.

At the time, the agency was going through a rough patch. In general, the culture was quite toxic. The company's leaders were at odds with each other, and this created what I can only describe as "camps" of employees who positioned themselves with one of the three founders. As you can imagine, this put most of the employees at odds and created a lot of distrust. The only thing that was consistently working across the company was the gossip machine. Things weren't much better with the tech group. My team had an "us versus them" mentality because "they don't get us."

This was a tricky situation for a newly appointed leader to walk into. I identified a company that had lost its purpose and direction, and a culture that had gone sour from constant frustration and disappointment. I was all the more intimidated after I went to a good friend and

colleague for advice about the situation and she said, "You should probably look for something else and jump out of that crazy salad." No help there.

Someone once told me that if you see it, it's your job. And I could see all the problems. I knew that while I couldn't solve them all, I could at least make changes within my team and hope that our changes would create a positive effect on the rest of the negative culture.

Faced with a difficult turnaround situation, I leaned into my strategic planning skills and did what any strategist would do first: analysis.

Current state analysis

You can't plan your way forward until you know where you're at. While I was clear on the general problems the team had (they were certainly eager to vent about them), I wasn't clear about why these things were happening and who owned the problems. I started my analysis by asking each person on my team questions, like:

- What are the current challenges to producing your best work?
- How do our processes and policies either help or hinder your work?
- Are you doing the work you love to do?
- Do you have all the tools you need to do your best work?
- How can I help you do your best work?

I documented the answers and highlighted common themes, and through that I was able to find some key insights behind most of our difficulties.

Future state and gap planning

The next step in my strategic turnaround process was to let the team dream a little. This was the fun part. I got my team together and led them in a visioning exercise where we got to think about our ideal team state. I asked questions like:

- What tools, technologies, and sites would you build if you had the freedom to do so?
- If you ran the place, what would you change first?
- What does the perfect work day look like to you?
- Talk about when you were the most productive you've ever been.
- What are your strengths and what are the strengths of your team members?

Armed with the current state and the team's ideas about

their ideal future state, I could then identify the gap. Ours was a wide one, but I was able to weigh and prioritize those initiatives that I felt we could feasibly change, that we would be willing to commit to, and that would have a positive impact on the rest of the organization. We identified three key areas where we could control and implement improvements.

1. **Team process improvements:** We focused on meaningful changes that would help us optimize our project process and cadence. For example, instead of a big, one-hour long Monday morning meeting, we instituted quick 15-minute stand-ups at the start of each day to align tasks with our strategic goals. We also got rid of the technologies we all hated and streamlined down to Slack and one other tool.
2. **Tech team camaraderie:** This consisted of simple, consistent things we could do to deepen relationships and build positive feelings about work, such as Friday afternoon celebrations at the bar down the street and daily 20-minute team walking meetings around the neighborhood.
3. **Inclusion of other teams into our work:** In order to break down the "us versus them" barrier, we needed to formalize engagement across team structures. I challenged my team to respond to "they don't get us" with ideas for how to help them. Each of my team members took a topic and created a slide deck and we hosted "lunch and learns" for the entire agency to teach them different topics we were interested in. This was a successful effort. After one session we got most of the account team to use Markdown, for example.

Turnaround like a software sprint

We treated each of these areas like software development epics on a sprint board. Each key area had individual tactics, deliverables, owners, and success criteria. I appointed a scrum master, and each person had work projects they were responsible for completing. Then, as we executed, we kept track of our progress. We held regular meetings to review our individual projects and discuss blockers.

So what were the results? Running our "culture betterment" initiative in this manner did a few things that I could not have planned for:

1. It aligned my team behind a clearly defined, clearly communicated, and well-structured purpose; which led to….
2. A team that became advocates for making positive changes in the organization and evangelists for our cause; which resulted in….
3. The rest of the company beginning to appreciate and adopt our positivity. The momentum we created with our sprinting led to changes across other teams as well.

I'm sympathetic to developers-turned-leaders who find themselves in the midst of challenging team culture situations. It can feel overwhelming, and the mountain can seem unscalable. But just like I leaned into my strategy background for the solution, dev leaders can leverage what they do best: problem solving. You probably already have systematic ways you troubleshoot a particular problem in code. See if the same framework could apply to your team. Do you assess, sketch, reverse engineer, test, or ideate? These same processes can apply to challenges within your team. I encourage any leader in a turnaround situation to leverage their experience, use it to reframe the situation, and empower their team to take ownership. The results can be deep and lasting, not to mention highly satisfying.

Comment:

Software Development is a Team Sport.

View post

Brink Insights

brinkinsights.com

@jaymethomason

Jayme Thomason is not only an editor for BuildBetter magazine, she has spent her 12-year career solving the most complex digital challenges for companies of all sizes. She has built several successful businesses, including software company, DivvyHQ and her latest venture, Brink Insights, where her team of technical experts helps software companies analyze their digital systems, uncover opportunities for growth and implement experiments to find core drivers of business.

Jason Taylor
/ CTO / Stackify

Is Your Team Designing for a Disaster?

Three Don't's and a Do

Development teams work at top speed, and the environment in which they work is demanding and complex. Software is no longer considered done until it's shipped, and there's been a tendency to view shipped software as preferable to perfect software. These philosophies, created in large part by agile and lean methodologies, celebrate deployments and meeting release cycle deadlines. But have our standards of working software quality been trumped by the rapid pace of delivery?

We practice Agile development at Stackify, and are advocates of the methodology as a system to build and deploy better software more frequently. What we don't subscribe to is the notion that the process we take to create better performing, high-quality software is more important than the software itself. If the process gets in the way of the product, it's time to re-evaluate it.

The beauty of a process like Agile is that you can modify it to suit your team and what's most important in your delivery. Here's a bit of what we've done to optimize, and some of the things we've learned along the way.

Don't let quality draw the short straw.

I've yet to see a project that doesn't have a problem when it gets past development and heads into the final push towards "done." Primarily, I see one of two things happen:

1. **Testing cycles are compressed or incomplete due to time constraints.** A major contributor to this is that code often isn't ready

to be tested until it's all ready. As the sprint burns down, more and more code tasks are complete but they've yet to be reviewed, merged, and deployed to test environments. The code is rushed through QA, and ultimately, issues are found in production. Fixing those issues robs time away from the next sprint.

2. **Testing doesn't get compressed, but it extends the sprint in order to be complete and/or fix problems.** In a scenario where sprints overlap (i.e. once dev is complete for Sprint A, developers begin picking up tasks for Sprint B), this has a domino effect throughout the entire schedule of releases. We commonly joke about this being a "death march," as it creates wider deltas of code diffs, more complex merges, and a general log jam of productivity.

We are no stranger to these phenomena ourselves. Like any other dev shop, we have testing tools that help out, running automated UI tests, unit testing around core functions, and automated/ manual integration tests of complex system functions. But, it's still really hard to get through everything we'd like, at the level of detail we'd like, in a reasonable time frame. There are two criteria I ultimately base a go/ no-go release decision on:

- **Confidence.** Have we accomplished our goals for the release while also improving (or at least maintaining) our application's overall performance, stability, and reliability?
- **Risk.** What have we changed, what can it impact, and do we fully know the scope and scale of that impact?

Throughout our development process, we use Prefix and Retrace to help us build confidence toward the next release and to assess how much risk we have. Our developers run Prefix on their development machines, finding and eliminating bugs, bad code patterns, and performance issues before they commit code.

Our developers, QA team, and management all use Retrace to look at overall performance and new and regressed errors in each one of our pre-production environments at each stage of our dev lifecycle. We know, from build to build, if we have introduced new problems and moved the platform forward or backward. It's a tangible measurement of the overall health of our release.

Don't fear a punch.

As Mike Tyson once said, **"Everybody has a plan until they get punched in the face."** You're going to get punched in the face. It's inevitable. There will always be a problem, an unexpected server or cloud failure, a critical bug uncovered, support requests, or an "urgent" need from someone else in the company to get something that isn't in the plan done ASAP.

If you know you are going to get punched, your process must plan for it. At a minimum, you must be leaving some capacity to deal with it. If you're doing it well, you should have an "expedite" lane on your planning board (along with an established process) to deal with items that come out of nowhere with a lot of urgency.

Be wary of the tendency to let too many tasks fall into the "urgent" category. It happens sometimes due to pressure from somewhere in your organization, or just because of fallout from a chaotic application. By its very nature, urgent work will always have higher risk and the introduction of an opportunity for shrinking quality. If it's all urgent, nothing is, and you're just sacrificing the quality of your product.

Don't be a process zealot.

In theory, having an Agile development shop sounds great. In practice, it can be great.

A trap that many fall into is trying to carry out a textbook implementation of agile, often times attempting to cram what really happens into what should be happening. But it's not a one-size-fits-all process.

Craft your Agile implementation around the way your business needs to deliver software. A great example of this is the concept of "scrumban" that many teams have started to adopt. The focus is on limiting the work in progress, but still having timeboxed sprints and releases.

SCRUM would dictate that you release at the end of each sprint. If this could have a negative impact on your customers, would be difficult to manage because of frequency, or for any other reason doesn't fit your needs, then simply modify the process to work well for your specific project.

Make agile work for you.

Agile is meant to help teams get on a cycle of continuous

learning and deployment, and if systems are not in place to check for success or quality, you'll have a lot of problems. Any amount of downtime or rework is too much for software teams and businesses that rely on applications as their primary revenue source. Efficiency, in regards to the software development lifecycle, should not be married to "faster" deployments, but stabler, higher quality deployments. Successful deployments should adhere to the rule of "working code." Too many times, "done code" is not "complete code," and then it turns into an emergency. Agile teams must build a better criteria for success to accomplish the company's objectives for the code, and not just adhere to a time-boxed deployment schedule.

Many companies wait until there's a problem to make an adjustment to their team processes and products, but we think time is better spent preventing problems and avoiding work that interrupts from the greater business objective. Prefix and Retrace are more than pieces of software, they're comprehensive app performance tools that works alongside your agile team to help you plan ahead, build proactively, and deploy without interruption or emergency.

There's nothing like a few emergencies to challenge your team's morale and focus. Agile is a great start, but building in support structures of purpose
and product means that your developers can focus on their code performance and your company can focus on building better products. It's a win-win for everybody.

Comment:

Stackify

stackify.com

@dotnetjt

/in/tig66208

Jason Taylor has worked in a number of high-growth business units centered around delivering Software as a Service. The experiences gained in those shops directly led him to Stackify, and those experiences help shape the product. Jason has led small- and medium- sized development teams through his career and is intently focused on delivering a great product while helping developers grow, learn, and realize their full potential.

Get published in BuildBetter

Subscribe for more BuildBetter

Stackify exists to increase developers' ability to create amazing applications as quickly as possible. We believe that the ideal development team, today and in the future, is consistently optimizing its output across the entire lifecycle of an application; from development to testing to production. Stackify's mission is to give developer teams easy access to powerful tools, which enable them to take the lead in delivering the best applications as quickly as possible.

If you're a developer, team lead, or architect, Stackify's tools were built for you. In fact, have two game-changing, code performance products no developer or dev team should ever be without:

Prefix

Prefix is a popular developer tool for finding and fixing bugs while you write your code. You have profilers and debuggers, but nothing is like Prefix, and it's free!

View the 60 sec. tour.

Retrace

Retrace is an APM tool built specifically for developers and dev teams. Our agent can install on pre-prod or production servers to make black boxes transparent for the coders. Try Retrace free for 14-days.

View the 60 sec. tour.

Build*better*

v2.1.JFM17

Publisher: Stackify, ©2017

Editor in Chief & Creative Director:
Max Hobbs

Managing Editor:
Jayme Thomason

Technology Editor:
Taylor Ford

Copy Editor:
Laura Drucker

Art Director:
Kevin McCoy

Production Manager:
Jennilee Tangpuz

Marketing & Promotional Partner:
Brink Insights

Stackify