PACERPRO

We Deliver

Always online, PacerPro delivers 9.8M case notifications a year and how our CTO still sleeps well at night

01 Introduction

Ken Mayer is PacerPro's Chief Technical Officer, which means that he gets the first phone call at 3 am when things go pear-shaped. In this white paper, he talks about why he doesn't get many of those calls:

He's built a development team and process that delivers reliability as a core feature. Mr. Mayer is 30 plus year veteran of the software industry; a sporadic blogger; an unpredictable tweeter; a recovering SCUBA instructor; and an erstwhile master of vessels steam and sail. Everything in the last sentence is absolutely true.

02 No one gives a damn about where the electrons come from until the lights go out

The number one core value of PacerPro, from the very beginning, has always been rock solid reliability. "You only get your customer's trust once," says Gavin McGrane, CEO, "and if you lose that, you'll never get it back." So we've built our product with reliability as a primary design constraint.

That's a very, very tough requirement. Most startups (as we were, back in 2013) write their code with duct tape and bubble gum. We write our code using a "Test Driven Development" (TDD) methodology, which means that we write the tests, first, before a single line of production code is written. We only ship code to production when all of our tests, including all of the other tests we've written over the years have also passed.

We run a lot of tests; thousands. We run our test suites dozens of times a day, sometimes hundreds.

I've been working in the software development industry since 1985. Back then, in the bad old days, production deploys were fraught with worry and prayer. Even so, way back then, I had a reputation for shipping bug-free code. Annoyingly so, since sometimes that required delaying the release until the code was really ready to go.

03 Stay focused on product & core competencies

We're a lean company. Infrastructure is not our product; I don't want to spend precious headcount budget on operations. These days, with everything in the cloud, PacerPro gets to leverage the expertise of some of the largest IT corporations in the world: Amazon, Salesforce, Google.

As CTO, I find "best in class" providers to deliver computing infrastructure for our product. I don't have to pay for a devops engineer or an expert DBA, on the other hand, I get my provider's expertise in their core competencies. So, for a fraction of the cost of a single engineer, I have scalable databases with redundancy, hot backups, and immediate roll forwards. All ACID, encrypted at rest, and privacy compliant. There's a security vulnerability in the base operating system of our database server? No worries, our provider delivers the patch, at scale to every server in their cloud. And due to our designed in redundancy, except for the courtesy notice on the security mailing list, we don't have a noticeable outage.

04 You're not paranoid if the world is really out to get you

There's an interesting side-effect to building cloud-based applications: You have to engineer with the expectation that any of your service dependencies can and will fail at any moment.

This is where an always-testing culture comes in handy. Part of our production test suite has what is sometimes called "enemy testing," where we simulate that something "bad"™ has happened.

We don't have to guess whether our application will run or not in the face of failures, because we have empirically proved it to be so. One of our web servers dies? No problem, we have 3 more running. Someone mentions us on Hacker News, our automated scaling system spins up more servers to handle the extra load. One of the PACER sites goes down, we'll trip a "circuit breaker" in software to prevent jobs from stacking up on the dead resource, until it comes back online again.

05 Commodity services are just another API

PacerPro is remarkably flexible when it comes to service providers. There's virtually no lock-in anywhere in our tool chain. This is because we have a relentless obsession with reliability. Our software is built to talk to services, not service providers.

Inside our service code, we'll have pluggable references to our providers, including all authentication and configuration information, so if (and when) we need to switch, we can quite simply change a configuration variable in our production environment and continue on around the failure. PacerPro generates over 50,000 individualized custom emails every single business day; we can't wait for a downstream outage to "resolve itself."

06 Privacy by design

There's a lot of chatter these days about security breaches, which is sad because, while a challenging engineering problem, it is by no means unsolved. The trouble starts with senior managers who don't know anything about security (and why should they, after all), but:

(1) they don't budget for it and
(2) their engineering staff does not or can not communicate the mission critical aspect of it.

Then security becomes an afterthought.

PacerPro does several things: Security design is part of every story (usually it is a non-issue, but we do check it). We try not to store any sensitive data, period. You'd be surprised how much you don't need. We don't trust ourselves to do encryption "right," so we, again, offload it to the experts. For example, we use Stripe.com to handle all of our credit card transactions, so never even see a credit card number. We encrypt the data that we must store. We have regular scans for code vulnerabilities and patch them as "Level-1" bugs (usually released the same day).

If asked whether all this extra work is really necessary, my reply is always, "Do you want our company logo to appear next to the headline, 'Hacked'?"

07 Building a culture of "writing things down"

No one is irreplaceable. Not even the CTO. Sometimes we joke about "bus counts," which is the number of people that have to be run over by a bus before the company can no longer operate. A bus count number of "Ken" is unacceptable (although hard to eliminate as much as I try).

So, we write down procedures in a company “run book.” We automate the heck out of anything that we do more than 3 times (3 is a magic number). We have a company chat channel where we can call an “all hands on deck” emergency, and because we a distributed company, we’ve got bi-coastal coverage.

Our engineering culture is “full-stack;” everyone has the skills to be familiar with any part of the product. Some of us are more expert at some things than others, so we use pair programming to “level-up” our team knowledge.

Pair programming is an “Xtreme Programming” technique where two engineers work on a single story at the same time. I sometimes describe it as “Two brains, one keyboard.” This is also another way we increase reliability. Pair programming may have a bum rap because you are paying two developers to deliver one feature, but empirical studies show that the code quality is much higher; enough so that the overall code cost (in time and dollars) is about a wash: But you get to higher quality, more reliable code sooner, for the win.

08 If you don’t measure it, it doesn’t exist

Another core part of our development process is *‘a priori’* support for metrics. We measure everything, from email delivery latency to database transactions per second. Again, not an afterthought. Sometimes we’ll only care about a metric for a short while, such as when we want to optimize a critical block of code.

Most of the code that we write is optimized for developer productivity, by the way. Between our own skills and Moore’s Law, 85% our code base is fast enough, as is. That way, we only have to focus on a few critical points in the software, instead of guessing.

Other metrics go onto our production dashboard that we distribute internally. We have scripts that monitor for out of bounds values and then send notices to the engineering and devops Slack channels. We used to have a PagerDuty account but we found that to be too heavy weight for us. Since we already listen to Slack anyway, it created less friction to our regular work process.

09 No, you can not have a pony with that

Gavin, our CEO, can attest to the fact that I’m a certified pain in the ass when it comes to building new features. That’s my job. Before we start designing, I ask a lot questions to sales, product and C-levels, “Would this be stupid for us not to do, in the next 90 days?” “Based on the feature, it is going to cost $10x THOUSANDS of dollars to build, test, verify, deliver and maintain. What’s our ROI over the next year.” “What’s our opportunity cost if we build this feature instead of that one?”

That keeps us focused. 90 days may seem pretty short, but our world changes so quickly, that 90 days from now, everything that we thought we knew about our business will either change or be called into question. Long term planning in this context is a fool’s game. That’s not to say that we don’t have longer range plans, but they are not specific implementation plans.

This is where traditional engineering and software development diverge: Building bridges rely on (among other things) the tensile strength of steel and the force of gravity. Neither of which is likely to change over the life of the project. We build software with the knowledge that we are modeling human behavior, which is subject to change (our knowledge of the behavior, not the behavior itself) without notice. So we optimize for rapid, constant change.