accusoft®

Moving from "Doing Agile to "Being Agile"

John Armenia
Director of Self-Hosted Solutions

Accusoft has been developing software for almost 30 years now. In that time, the way software is delivered has changed. Because the way software is delivered has changed, the way software is built has also changed.

Back in the early days of software, once you were "code complete," there was often several weeks or months of additional overhead to get your software to your customers. Things like mass producing the floppy disks or pressing the CDs, printing the paper manual, and having all that shrink wrapped and shipped to your customer. Because there was so much overhead in getting your software to the customer, release cycles were very long, and making sure what you delivered was as bug free as possible was very important (was patching even a thing back then?).

Today delivering software has become much easier. Engineers write the code, documentation, and tests all at once, and with a quick click of the mouse the code can be deployed and available for customers. This reduced friction is what enabled the Agile Development Methodology.

The Problem

When I took over the PrizmDoc product group back in December of 2016, much of the way things were being done still reflected the development practices of the past. Long release cycles, deadlines being pushed back so one more feature could be added, features developed that didn't solve the customer's problem, hundreds of hours of manual testing, and on and on. All of this despite the fact we were "Doing Agile." We had daily standups. We had two week sprints. We had a backlog. We had Sprint Reviews. No one, however, would argue we were agile.

The Solution

My plan consisted of five key points:

1. Set expectations
2. Identify key metrics
3. Get the right people
4. Execute toward the metrics
5. Reward the right behaviors

These are all things any good plan consists of but it was important to have that plan and follow through.

Set Expectations

The first area I had to address were the expectations of the stakeholders. I didn't have a magic bullet and it would take time before we would see a return on the investment. I warned them that new feature development would slow down (it did). Team velocity would go down (it did). We all would celebrate successes, no matter how small (we did). Once I had the stakeholders on board, I had to get the development teams on board as well. After all, if they aren't on board, nothing gets done.

I worked with my development managers to set expectations with the teams. Previously, we had issues with two teams working on the same feature, both being done, but when we put the parts together they did not work. This was because virtually all of the product level testing was being done after code freeze. We agreed to change the definition of done to include the story is tested at the product level. We also agreed velocity would be reduced because some of the team members would have to learn how to use the product.

The second item we agreed to was more frequent releases. When I presented this to my teams, they thought I was crazy. They thought I was crazy because we had 6 week release processes. The entire product had to be regression tested manually. Inevitably some critical bugs would be found and fixed, then the processes would begin again. I pointed out that more frequent releases will allow us to optimize our process. Additionally smaller releases mean less risk of a catastrophic bug being found during testing. This led to the next step, good automated testing.

We agreed to focus on creating a build pipeline with tests that are meaningful. If a test isn't useful, we will get rid of it. If it's failing, we will fix it. No longer will we say, "We expect that test to fail." While the teams agreed with this change in theory, they felt the main reason they weren't already being successful was because they weren't given the time.

To resolve this concern, we agreed to follow the Scrum process in which the team sizes the story and those size estimates should include the effort to test. We agreed that the stakeholders would never tell them how much effort a story should take, but we would be able to question why their estimate is what it is and perhaps change the scope (before pulling it into a sprint) if necessary. It did take some time to build this trust, but we have been successful here.

Finally, like with the stakeholders, I emphasized we would celebrate success no matter how small it may be at the beginning.

Identify Key Metrics

Because many of the PrizmDoc customers integrate our software into their own product, there can be a significant delay between our release and when we start getting feedback from our customers. To help mitigate this, we identified several leading indicators that could help us know if our process changes would result in better experiences for our customers. These indicators are:

- Average Bug Score (our internal scoring system for rating bugs ranges from 1 to 25
- Number of critical bugs found (Bug Score > 15)
- Stabilization effort (how long from the time we tag RC0 until we release)
- Number of Release Candidates (RCs)
- Test stage run failure percentage (what percentages of our test runs fail)

As previously mentioned, we felt if these metrics were going in the right direction, we would be providing a better product and more value to our customers all while making our teams more efficient. To confirm this, we identified three trailing indicators that would help us understand the actual impact to our customers. These indicators are:

- Number of support tickets opened (if customers are calling support, our product is either too hard to use, or too buggy)
- The bug score of bugs reported by customers (if customers are reporting bugs, we want them to be less critical)
- Development Team Health Checks (if the development teams are more efficient and doing what they feel they need to do, we should see moral improve - we used the Spotify Squad Health Check as model)

Get the Right People

When I started working with the PrizmDoc Product Group, the QA members of the development teams were tasked with manual testing. In order to achieve our goal of more frequent releases and improving the quality of our product, we were either going to have to increase our QA staff several orders of magnitude or find ways to make our existing staff more effective.

We made two decisions. First, we would start to move the QA team members from a Quality Assurance model, where they are responsible for ensure the product is of a certain quality, to instead a Quality Assistance model (we based this on the Atlassian model). The second decision was to focus on coaching our existing QA members on writing automated tests; and when bringing any new members onto the team, we would look for them to excel at writing and designing automated tests.

Execute Toward the Metrics

In order to accomplish our goals, we had to ensure we were executing toward the metrics we had previously identified. We spent approximately 150 hours a month maintaining pipeline failures, a duty we affectionately refer to as "Stupid Robot Duty". When it came time to release, we have release postmortems that helped us identify the biggest pain points of the process so we could prioritize working on the solution at the beginning of the next release. The development teams sized the stories they were working on to include testing effort and the stakeholders kept their promise about discussing scope and not effort or time.

Reward the Right Behaviors

The final piece of the plan was to ensure we were rewarding the teams for the right behavior. We do this several ways at Accusoft ranging from giving someone a "Thanks!" to nominating them for an "Above and Beyond" award. After our first successful release, we took the teams out on a Top Golf outing where they got to kick back and play some golf themed games. To wrap up the successful year, we threw the first of what has become an annual bar-b-que for the entire engineering team hosted by the PrizmDoc Product Group.

In the next post in this series, I will go over the quantifiable results we've had so far.

The Results

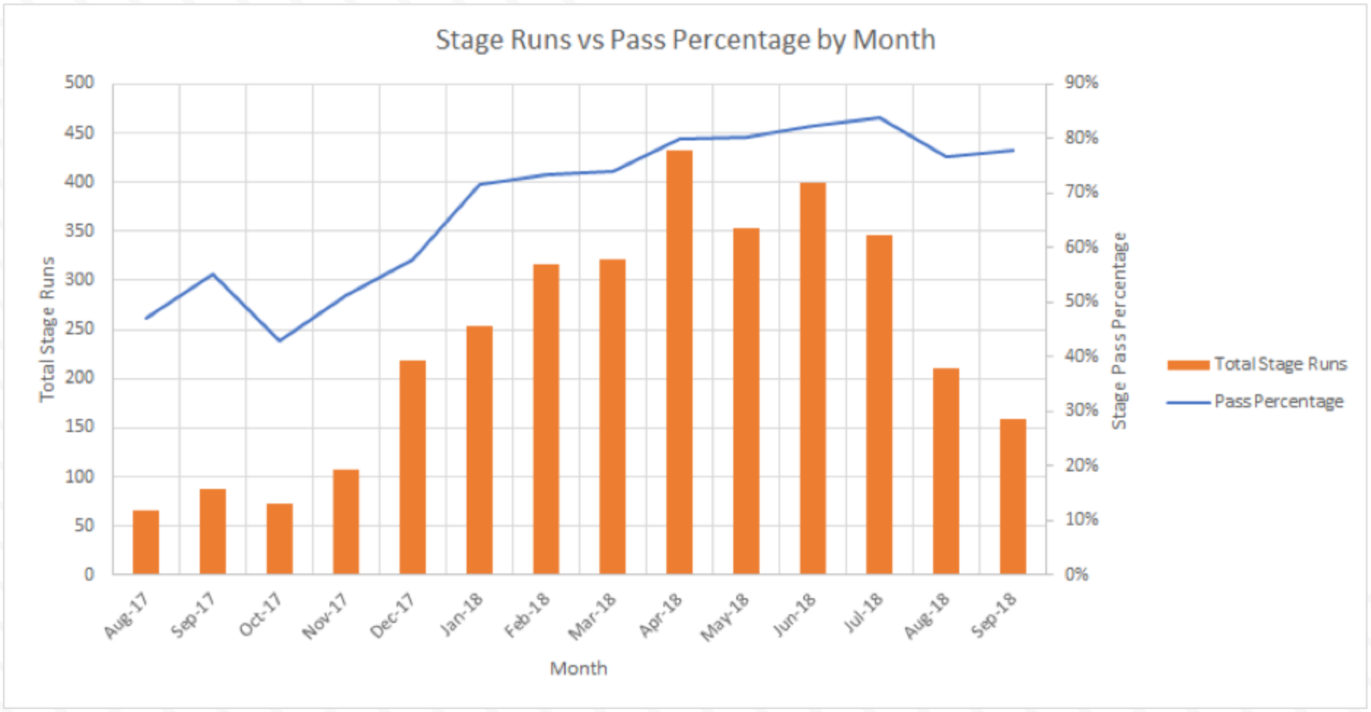
Faster Feedback

One of the driving goals of Agile development is to get feedback from the stakeholders and customers faster. Part of this involved shorter release cycles. Another part of this is about communicating with the customers. To enable better communication, our product management team created a Customer Advisory Board (CAB) that meets quarterly. During these meetings, they will review the roadmap, show prototypes of potentially new features or products, and gain an understanding of what our customers want from our product.

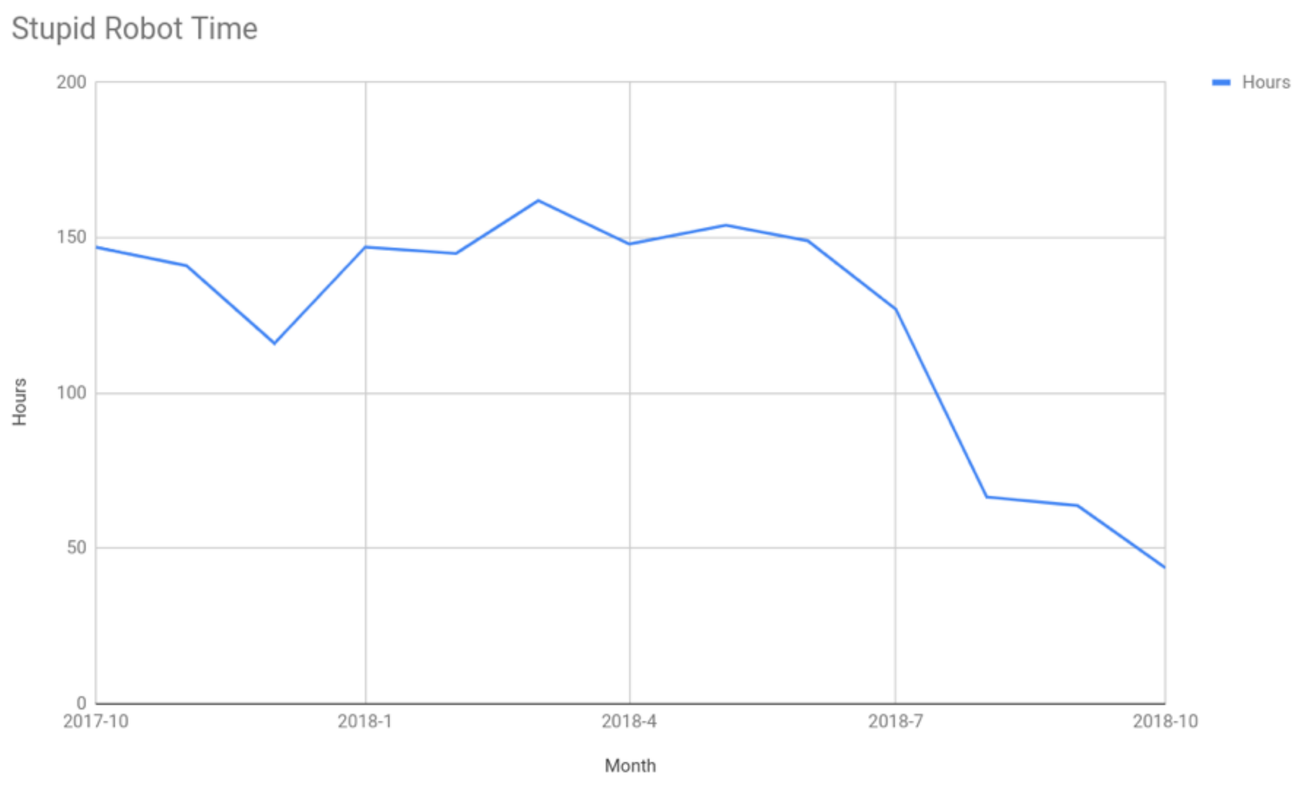
Additionally, now that we have an automated testing pipeline in place, we can deliver a beta build to a customer at any time. This enables us to get their feedback on a new feature they may have advised us on or help validate that we resolved an issue they were previously having.

More Efficient

A premise of the Agile methodology is that a team will provide more value to the stakeholders as they come together as a cohesive team. Additionally, the team's velocity should increase as their processes get more efficient. One of the areas we targeted for improved efficiency was our "Stupid Robot" process. This was job of reviewing failed tests and doing root cause analysis. We attacked this problem from two directions. First, we fixed issues in both the product and tests that were causing failures as they came up. This resulted in more consistently passing stages in our testing pipeline. In the graph below, the blue line is the pass percentage by month of all our stages combined. The orange bars are the total number of stage runs. The reason the number of stage runs is decreasing is because as stages become more stable, we are moving them to our nightly pipeline to free up resources for other tests.



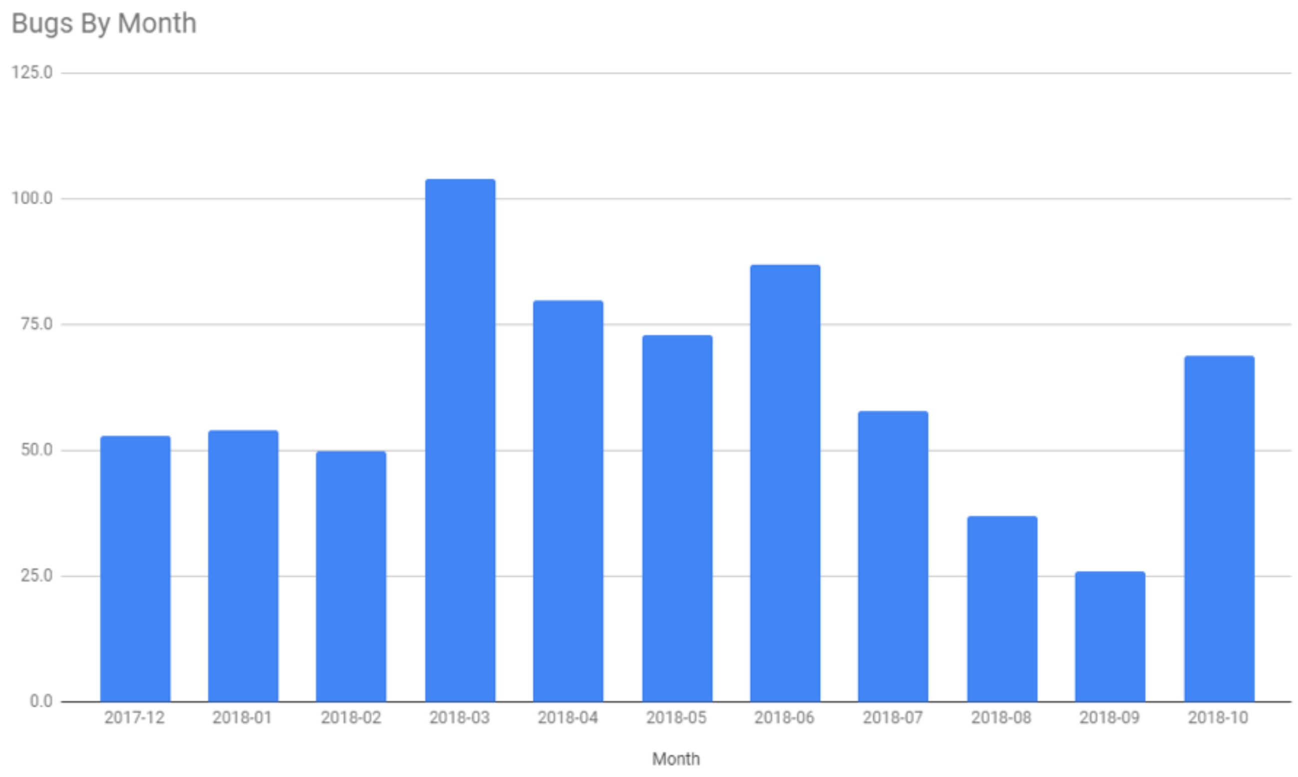
Second, we updated our internal tools to allow us to more efficiently debug and perform root cause analysis on test stages that fail. This has resulted in a significant drop in the number of hours we spend on or “Stupid Robot” process.



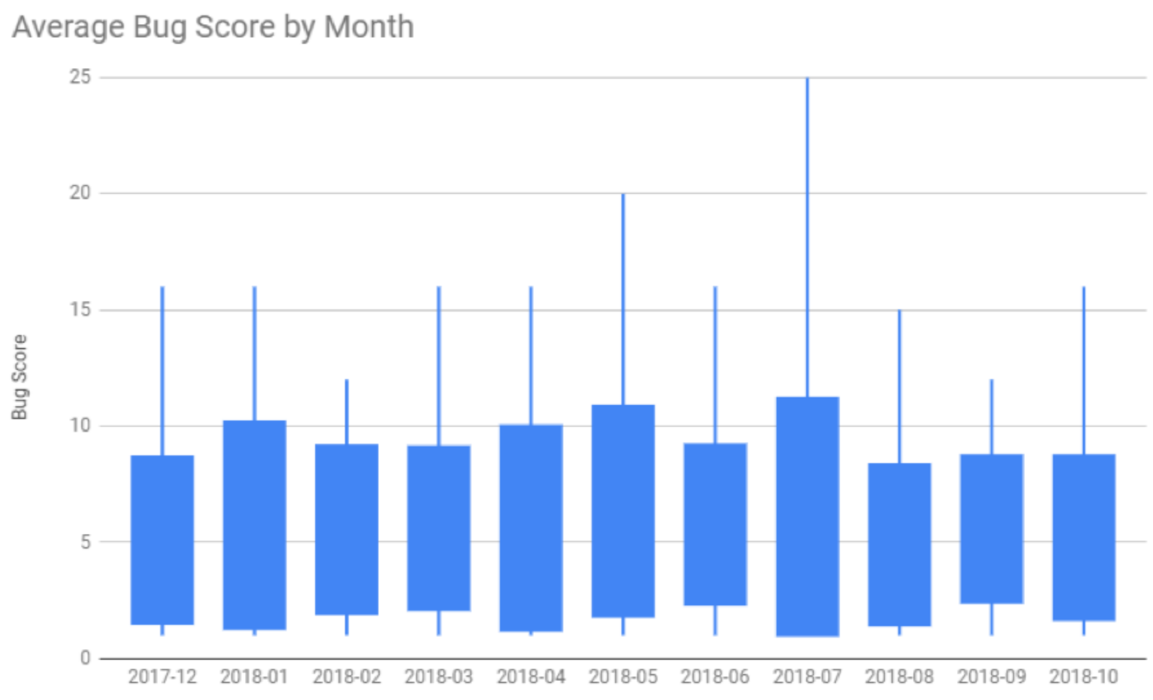
Finally we focused on reducing the number of business days it took the team to publish a new release. Like I mentioned in a previous blog post, many of my team members thought I was crazy when I proposed this. We still have a ways to go here, but we are in a much better position than we were two years ago.

Quality

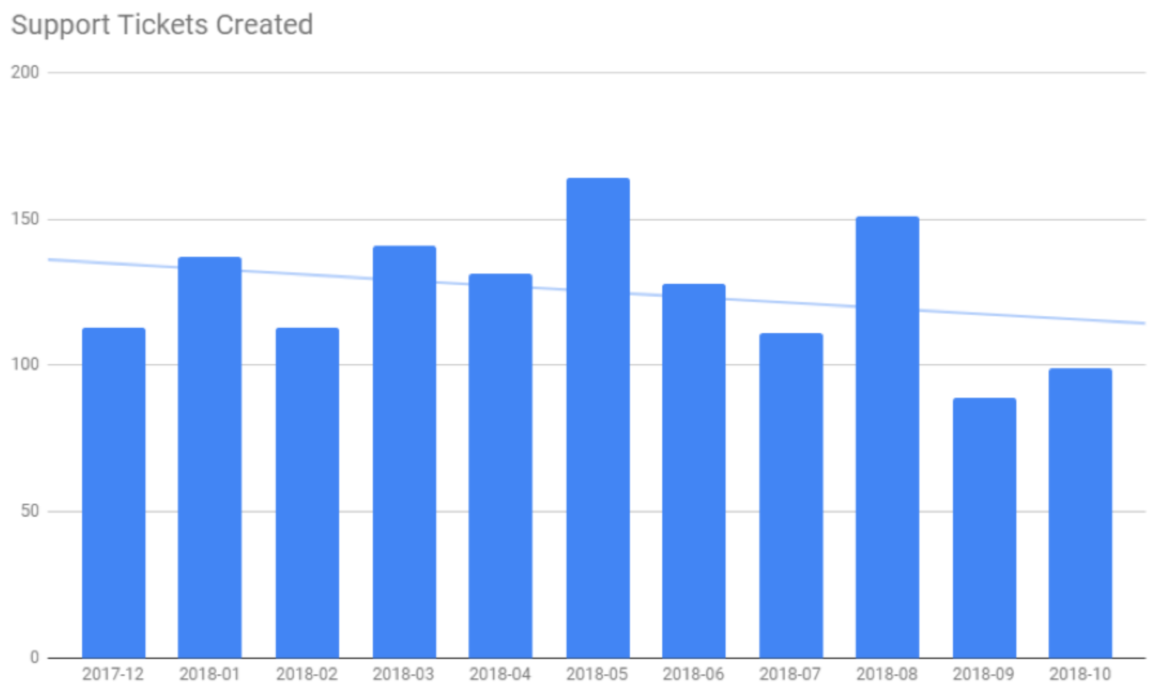
As we expected, the number of bugs being reported both by engineers testing the product and customers using it have gone down. Why the spike in bugs around March of 2018? That's when we started ramping up the number of production builds going through our test pipeline.



Additionally, we are seeing a reduction in the reported bug score. In this chart, the top thin line is the maximum bug score reported, the bottom thin line is the minimum bug score reported, and the thick bar is one standard deviation around the average bug score reported. As a reminder, our bug scores range from 1 which is very minor, to 25 which is the most critical.

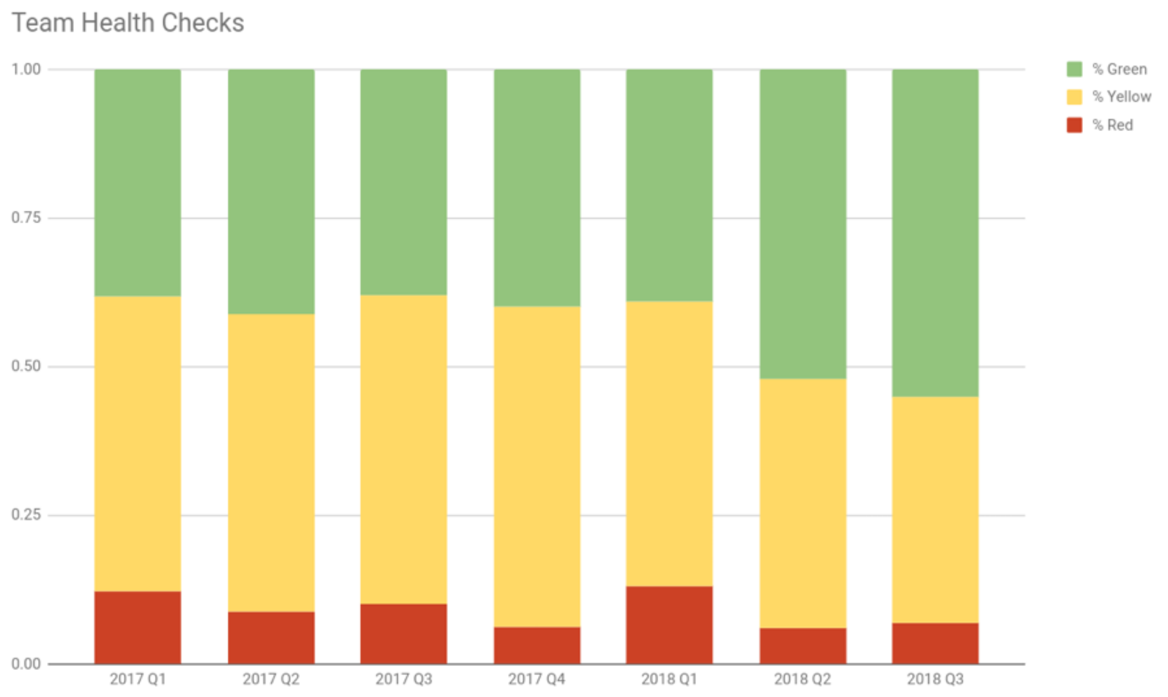


When we started this project, we expected the end result to be fewer incoming support tickets. At first, this graph doesn't seem to indicate much success. But considering there is a downward trend despite the product growing by 14% year over year, we consider this a success.



Team Health

A happy team is a more productive team. Anyone that has spent any time with developers knows that they hate dealing with technical debt, and hate doing anything other than writing code (well, most of them anyway). Given that premise, we expected to see team morale improve over time as we reduced the amount of technical debt and took more and more of the manual testing requirements away. This graph shows the percentage of "happy" developers in green, "okay" developers in yellow, and "sad" developers in red. As you can see we saw the percentage of "sad" developers move into the "okay" category at the beginning, and now we are seeing more of the okay developers becoming "happy."



Final Thoughts

There are certainly areas we have more room to improve, but there has been a dramatic improvement in the product group's moral, productivity, and customer satisfaction. None of this would have been possible without such a talented group of engineers and executives. I love that I get to work with them every day here at Accusoft. By the way, if you're interested in joining one of our teams, we are always hiring. You can check our open positions on the Accusoft careers page.