What would you say you do here? Reframing your Role in the QA Field

Jeff Sing

jeffsing@iterable.com, jeffsing@gmail.com

Abstract

The job of a QA Leader is to create a vision of your QA organization and engineers' roles at your company, but the struggle is that business needs are constantly changing. For example, do we even need a QA role anymore with advances in tooling and AI or with a cheaper headcount outside the US?

To address this, I want to reframe the role of QA in the field, not as testers or developer tooling but as engineers delivering QA as a Service (QAaaS). Running a QAaaS identifies the areas of needs that engineering lacks, highlights why the work is business impactful, and why the ROI of keeping this organization is essential. I will talk about how to build a pipeline of projects (technical tooling, developer enablement, quality process, governance system, culture shaping), justify value with telemetry/metrics, and the importance of marketing the concept of QAaaS to leadership and your peers.

Biography

Jeff Sing is a Quality Leader who has been in the testing industry for over 15 years. During this time, he has built automation frameworks and test strategies and executed quality initiatives for fields such as medical devices, infrastructure security, web identification, marketing tech, and experimentation and progressive delivery.

Jeff is currently a Director of Engineering at Iterable, where he leads the Quality Engineering organization in orchestrating their quality control plan utilizing a combination of automated testing and implementing QA procedures. He also acts as Iterable‘s customer experience champion to ensure Iterable remains the world's leading customer engagement platform. He has also built and led Iterable‘s Engineering Operations Team, which runs the services and programs to improve effectiveness and productivity systematically across the engineering organization as it scales.

1. Introduction

Recently, I spoke to an acquaintance who was struggling with his journey in the QA field. He was worried about his role being replaced and wanted to see if he should transition to a more technical testing role. The more I thought about his question, the more I realized that reframing his role as a more technical role wasn’t necessarily going to provide him with more job security. If our roles are only to test and validate, what prevents our leadership teams from outsourcing testing to a third party or transitioning testing to developers? Contracting testing overseas is significantly cheaper than hiring a full-time tester. Many companies have eliminated their testing teams entirely and shifted testing responsibilities to their developers. On top of this, multiple third-party vendors sell low code testing tools or AI testing tools that

promise to make dedicated testers obsolete. So, if I'm the head of engineering and need to cut costs, why would I employ dedicated testers?

The answer would be that a quality program does more than just testing. They are building the solutions that ensure your engineering organization successfully delivers the value proposition your customers have paid for. These solutions might be technical or process-oriented. Your quality program should also be able to define what quality looks like in terms of business impact with KPI and be able to measure it. Utilizing these data points, your QA program should constantly adjust to deliver and build upon a quality culture.

The good news is savvy QA Programs are doing a great job of these things! The problem is that most don't have a great strategy to ensure their engineering leadership team acknowledges this value. If the Head of Engineering only sees his Quality Engineering team as an organization that delivers testing, then when another service provider (vendors, third-party contractors, or AI) comes and can claim to disrupt testing (do testing cheaper, better, or faster), the in house Quality Program suddenly becomes expendable.

How, then, should Quality Programs ensure that their services are utilized and valued? **This problem is actually similar to the problem faced by all software-as-a-service (SaaS) companies with their customers!** They generate revenue when customers utilize their services. However, if the customer doesn't find value, they will churn. This could be because the customer isn't getting their desired outcomes, weak value from this service, or they can find a cheaper competitor. In the same vein, we should think of our quality programs as a quality service organization. We win when we deliver our product (Quality) and the customers renew or uplift (utilize what we offer, maintain our teams, or even scale the quality program). We lose when our customers decide we aren't the right solution, and they churn (lay off QA or reduce the scope of the Quality program responsibilities).

So, how do we reframe the role of our Quality Program to include work that isn't traditional testing that creates the value proposition that we are an essential organization? Going back to my analogy of SaaS companies, let's see how SaaS companies do this:

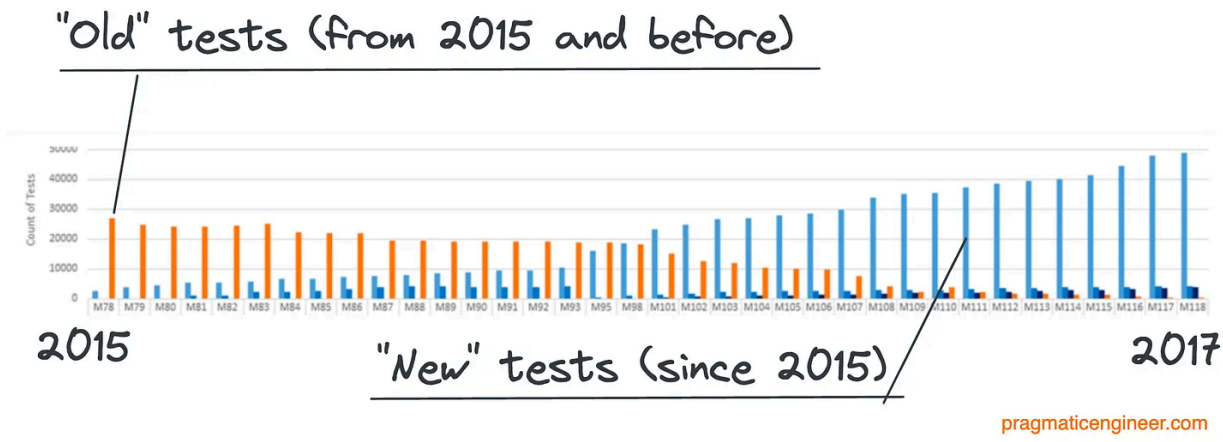
- Be able to understand their customer's needs and how our solution helps them be successful + ability to scale this solution with our customers as they scale (Product Management)
- Be able to support their customers when they have issues (Customer Service)
- Be able to show the value of their product or services (Account Management)
- Be able to renew and upsell their services (Sales)

The following sections will describe how the above Business Units in SaaS companies execute their strategy and how this translates to what a Quality Program should incorporate to deliver value.

2. The Problem: Are We Building the Right Quality Program?

Amongst my conversations with other Quality Engineers at different conferences, meetups, and co-workers, there is high confidence that we are performing tasks that will surely answer the question, "*Are we building the product right?*" We know how to find bugs, build automation, and develop sound testing plans. But what are we doing currently that answers the question, "*Are we building the right product?*" How do we determine if the work we do in our Quality Program fits for purpose?

In 2014, Microsoft laid off most of its QA organization. "*We combined the dev and test orgs into a consolidated 'engineering' org. For the most part, we eliminated the distinction between people who code and people who test. That's not to say every person does an identical amount of each, but every person does some of everything and is accountable for the quality of what they produce. We also set out to completely throw away our 10's of thousands of tests that took 8 years to create, and replace them with new tests that were done completely differently.* [1] "



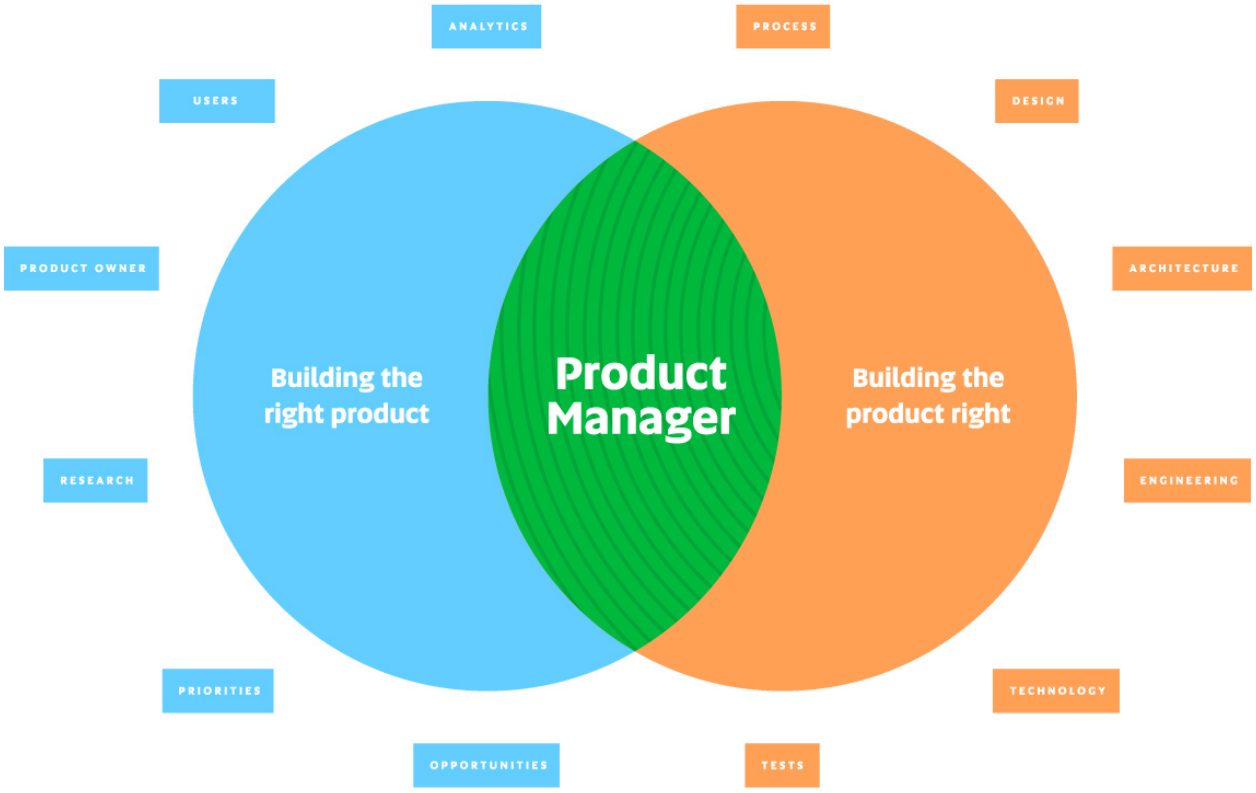
In 2 years, almost all "old" tests from when test was separate from dev, were gone. The new tests became more granular as well.
Source: Microsoft Dev Blogs

Looking at this decision, we see that the Microsoft QA organization invested in a product that wasn't necessarily the right solution (end-to-end test). Because of this, their roles were easily replaceable (developers can write tests).

If we want to run a successful quality program, we need to be sure that we aren't just "building the product right" but also "building the right product." To do this, we need to ensure that we are building a pipeline of projects (technical tooling, developer enablement, quality processes, governance systems, and quality culture shaping) that justify the costs of having one.

How do most SaaS companies accomplish this? This falls under their Product Management organization. Let's see how we can take some of the techniques from Product Management and apply them to our Quality Engineering organization.

2.1 Product Management for Your Quality Services and Programs



Building the Right Product vs Building the Product Right. Source: Mojotech product blog

Product Management is responsible for defining desirable, viable, feasible, and sustainable solutions that meet customer needs and supporting development across the product life cycle.

So how does the Product Manager organization identify the work that is viable, feasible, and sustainable to meet the customer's needs? Here are some ways:

- Market Research: Identify customer needs and market trends
- Product Strategy: What do you want to achieve, how do you plan to get there, and which products align with our core goals
- Roadmap Development: Outline the direction, priorities, and progress of a product over time
- Product Lifecycle Management: How to handle your product as it moves through development, introduction, growth, maturity, and decline.

For each of these, I want to share takeaways on how we could use these in our quality program to launch the most relevant services.

2.1.1 Market Research

A Senior Product Manager once told me his favorite interview question he liked to ask PM candidates. It was "*How do you design an elevator?*" The number one thing he wanted to see before anything was designed was for the candidate to ask, "*What is the building, and who is actually using the elevator?*" The idea is that you shouldn't be building products or services unless you understand whom you are building

them for. In the same sense, who and what is your quality program building for? As QA, we often assume that our customers are business customers. But have we considered our internal customers, the engineers? Do we understand their needs? In my example above about Microsoft, "*Full testing would take the better part of a day to run, many more hours to "analyze the results" to identify false failures, and days or weeks to repair all the tests that were broken due to some legitimate change in the product. (...) Two years ago, we started on a path to redo testing completely.* [1] " The Microsoft QA team's product strategy was to churn out as much end-to-end test coverage as possible, but in the end, this wasn't what the engineers needed, and eventually, all that testing was replaced. Before investing energy in a strategy, have we interviewed our developers to understand what they are looking for in a quality partner?

One example I can give of understanding our customer use case before building the solution was a case back at a smaller startup I worked for. Our Feature Flags management was out of sync from staging to production at this company. This affected quality since we were testing a different product in our testing environment versus (flag on) what was out in production (flag off) due to the disparity of feature flag states. Leadership wanted to invest in a third-party tool to manage this or try to build some tooling to manage the flag states, but both solutions would take quarters to implement or were costly. Meanwhile, our builds became increasingly flakey due to needing to know what state of the build we were validating against. Talking to our developers, most of them mentioned that it would be simpler to track states through Jira and have QA modify the testing config files to reflect what was being changed. This was a bit of added work to the QA team, but taking on the ownership of Feature Flag states and maintaining our environments instantly allowed us to reduce flakey tests. By researching what our Developers really wanted, our QA program saved our company money, improved engineering bandwidth, and allowed QA to take on some work that had an immediate impact, even though it wasn't a traditional project a QA program would normally do.

2.1.2 Product Strategy

As a Quality Organization that provides services, having a product strategy ensures that the services we provide will actually achieve the goal the organization set out to achieve. For each product we build (for example, testing framework, new process, testing artifact, etc.), have we detailed our outcome and why the work done will achieve that outcome? Product strategy entails diving deeply into these two questions and ensuring what we create achieves our goals.

For example, if our service is end-to-end testing, I often hear that the goal is to provide high test coverage. However, at Iterable, the end business goal is to have very little disruption to our customer's experience and not high test coverage. I was asked once by a new engineering manager what our end-to-end testing feature coverage was. He was beyond shocked when I mentioned it hovered at about 30%. However, looking at telemetry for the high amount of traffic Iterable receives over our features, we have very few customer-reported bugs. We can maintain this by doing two things well. We utilize Heap.io to give us data on the most trafficked portion of our application so we know that the most critical areas have strong end-to-end coverage. The second thing is release management, and part of our Quality Engineers' job is to create a checklist that, before we release, we have monitorable dashboards and a gradual rollout strategy (Feature Flags or Cluster Deployments). If a defect is released, we can quickly detect and roll back that defect. In my example, our quality programs' product strategy aimed to ensure customers had a good experience (low bug report rate) and did not have high test coverage.

2.1.3 Roadmap Development

One of the most critical behaviors of a successful product manager is to be able to determine how complete a product needs to be before it's ready to ship. The capacity to manage scope while meeting

deadlines is critical for success. Sometimes, the time to market will trump feature richness. In the same lens, if we build tooling, sometimes a quick proof of concept to bake with developers is good enough. I once had an engineer tasked with migrating us from a legacy testing framework to a newer one. His estimation would be two-quarters of the work, but I challenged him to partner with a scrum team that would use this tool and build out only three supercritical use cases that could be delivered in half a quarter. The result wasn't fully production-ready (it needed to be manually kicked off to run, there were not a lot of tests, there were no dashboards, etc.) but could be immediately run to gain some validation for critical use cases. Because it was adopted so quickly, we got a lot of feedback that ultimately changed our requirements for this tool. Today, we've gone back and added a lot of functionality, including full integration to our CI/CD pipeline, multiple integration libraries that support other scrum teams, and more test cases. These features were used by 80% of our company and have a positive sentiment with our developers. By ensuring we adopted the pilot and generated feedback quicker, we built a tool that was more used across engineering.

2.1.4 Product Lifecycle Management

One of the most essential directions a product manager needs to consider is the lifecycle of their product or services. They have to manage the product launch and ensure its adoption and growth. While the product matures, what tweaks are necessary to keep it relevant? Finally, when it is time to sunset the product, how can it be done gracefully? Any process or tooling we build should be treated in the same way. When we launch something, do we have a strategy on how we want the rest of the company to adopt it? Are we measuring the growth in usage? Did we account for the effort we need to drive usage across engineering? At Iterable, we launched a new quality initiative requiring engineers to fill out a release checklist to ensure a successful launch. However, we didn't create an easy way to track compliance and governance. Because of this, many teams just ignored the checklist and deployed in noncompliance. In the end, we didn't achieve our goal since we didn't have a strategy for adoption.

Another example was when we had a legacy testing framework running in CircleCI, which cost us extra credits. At the end of the day, we realized the framework wasn't giving us that many benefits for the costs, and we retired it. If we had strategized our sunset or when we should retire a framework, we would have saved the company some money and sped up our deployment time. Having a clear lifecycle management of all the Quality products you enact is critical for its long-term success.

3. The Problem: Engineers Don't Trust Our Work

A few years back, I read a great article by Matt Eland called Partnering with Quality Assurance. This quote always makes me laugh because of how accurate it is: "*I've worked with some very devoted quality assurance (QA) professionals. I've also worked with some extremely talented developers. But when you put the two of them together – look out! When QA and development interact, sparks tend to fly, arguments start suddenly and swiftly, and things devolve into chaos.*[2]" The work our QA Program produces and the services we provide hold little value if our internal customers and the engineering team have low trust or use for them.

In the same way, how do SaaS companies deal with customers who do not trust their product or are struggling to utilize it correctly? If you are using a SaaS company product, and it isn't working as expected, you will create a ticket or reach out to your customer service manager from their customer service organization. This Customer Service organization aims to help its customers utilize the company's products and services. In fact, customer service is a critical function of an organization. One study found

that 7 out of 10 customers stopped doing business with companies because of poor support. Look at the following stats from a study on Slideshare [3]:

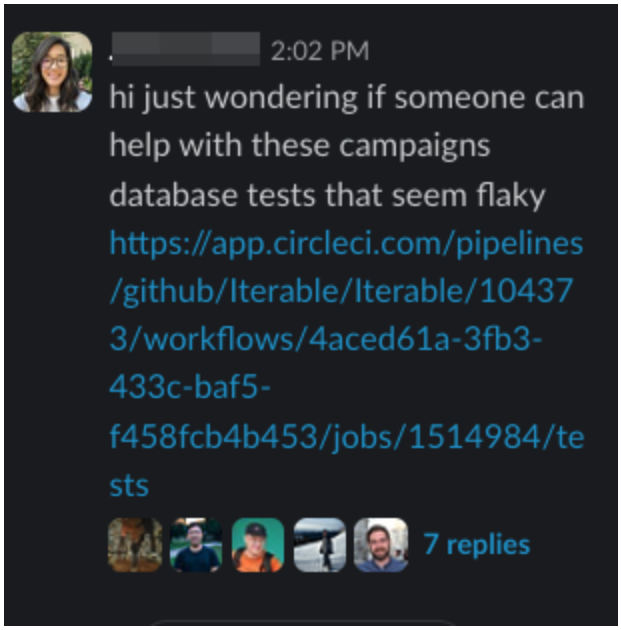
Looking at these stats, it becomes clear that we need to do more outside of testing to ensure our engineering customers are getting their perceived value. Managing this expectation and support will help guarantee that the engineering organization as a whole will have a higher value proposition of what the quality organization is providing to the engineering organization as a whole.

3.1 What are some examples of customer support in our quality program?

Here is how I have employed a few different services to provide support to our engineering team and leadership:

3.1.1 Quality Rotation

Each quality engineer at Iterable is in an on-call queue and goes on-call for a week at a time. During this on-call rotation, they are tasked with triaging flakey tests, triaging broken builds, responding to incidents to help engineers, and attending incident retrospectives. Even though this can be time-consuming, it gives engineers confidence that the Quality team understands their pain points with the testing infrastructure and gives quality a voice during incident management. It also ensures that when something isn’t right, our engineers feel like they can reach someone in QA quickly to start a resolution.



3.1.2 Open Ticketing System

We utilize an embedded QE model at Iterable, which means that only some teams have a dedicated QE. However, we do not want engineers without a dedicated QE to feel like they aren't supported, so we utilize Jira to allow engineers to request testing support. We have dashboards that show open tickets and how long they have been open and assign QE to help. When we first launched this system, we spent a lot of time advertising it, and now we see tickets created regularly, so we definitely know we have strong adoption.

3.1.3 QA Office Hours

Another service we provide to support our engineers is open QA office hours for engineers to discuss anything testing-related. We get questions about test plan creation, pairing on writing automation tests, and building infrastructure discussions. This open forum is nice to have, and some of the discussions can bubble up to leadership to drive higher-level engineering initiatives.

4. The Problem: The Quality Engineering Program Isn't Perceived as Essential

As a leader of a Quality Program, we must justify the cost of retaining our Quality Engineers and showcase the value they bring to the company. This challenging task is summed up quite well by David Caudill in his article Maybe Getting Rid of Your QA Team Was Bad, Actually. Mr Caudill writes:

"*The end result of this was that the slowest part of software delivery is testing. Since testing is why continuous delivery exists, that should have been good enough. Yes, we can make our tests faster, more automated, parallelized, etc. But when the highest value activity of a given practice is the bottleneck, you're optimal. You have achieved "the best possible problem".*

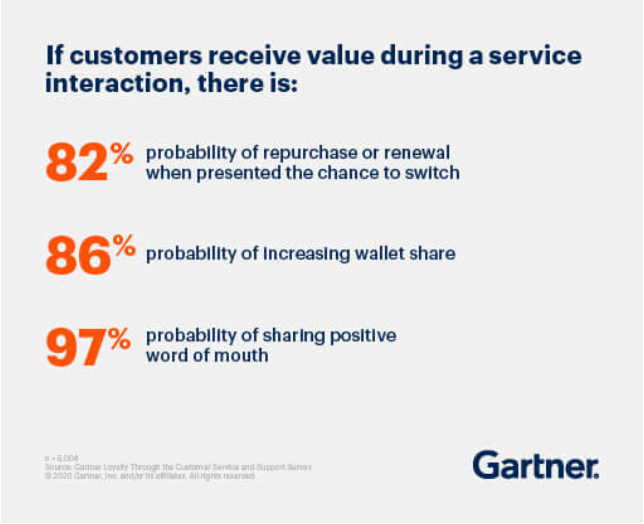
Those habits and behaviors of optimization didn't stop there. We kept on chopping. We squished our integration and end-to-end tests down to unit tests to parallelize. At the personnel level, we pushed out

anybody whom we believed could not code, indiscriminately, at the function level. We decided that testing might not be the bottleneck…the QA team was. The industry began to treat the people in these roles worse and worse. Expectations for them went up, salaries went down(while everyone else's seemed to be going up!), we contracted the role, we offshored it, pretty much anything we could do to try and stop employing QA Engineers. [4]"

The problem here is when Quality Programs tie their value proposition to testing, and if the engineering organization doesn't see the value, or worse, sees the Quality Program as a detracting value, it becomes a massive target for the chopping block. But in reality, the Quality Program does much more than testing, and its value goes beyond the testing pipeline. How do we show this value proposition? All SasS companies have an entire organization devoted to this very task, the Account Management Team. Let's see how they accomplish this.

4.1 Account Management for Quality Services

One of the most important organizations in any company is the account management team, which ensures that the customer is getting serviced well to increase their consumption of our product/service, maximize retention, and upsell more opportunities. By helping the customer understand the value they are getting from utilizing your service, account management ensures that the customer continues to use our services and opens up opportunities to use new services. Gartner [5] found that 82% of customers will renew when presented with a chance to switch if the company engages with them and performs value enhancement activities, which makes them feel like they used the product better and are more confident in their purchase decision.



As a Quality Program, we want to do something very similar. We want our engineering leadership team and engineers to understand the value they are getting from our Quality Program. If they understand the stories and see the telemetry of what is going well, they will be more likely to use our tooling or process and invest more in the Quality Program. Here are some ways that I've used to show the value of the Quality Program:

4.1.1 Monthly QA Reports

Every month, our Quality organization publishes a monthly QA report that contains our quality KPIs (Number of escaped bugs, Change Failure Rate, Number of incidents, Build Pipeline health, etc), success

stories that highlight what scrum teams are doing well, or failures and deep-dive root cause analysis over what happened. The goal of the report is twofold:

- Highlight Wins - Example: Our end-to-end testing framework caught four major defects, which prevented incidents
- Direction for Losses - Example: We had a new incident due to a change in the reporting page. We discovered that our test coverage was lower there, so we are adding more testing and monitoring so that this won't recur.

The goal is for each Quality Engineer to present the report's highlights to their scrum teams and for me to deliver this during our weekly engineering leadership meeting so the organization as a whole understands the value the Quality Program is delivering.

4.1.2 Demo Days

Every quarter, our engineering organization has a demo day. The QA organization always sends a member to demo something that the team is actively working on. When we demo, we utilize this template:

1. What is the tangible outcome of this project in relation to our engineers or customers
2. What impact will our customers feel once this is complete
3. What was the effect if we didn't do this
4. How do we measure the effectiveness once this launches
5. The technical portions underlying what we built.

The goal of presenting our demos using this method is to clearly communicate the value of what we build to our customers and make it clear that we are tracking how much our features are being adopted.

4.1.3 Quality Champions

One of my engineers developed this concept, and I love it. Once a month, we identify engineers who are doing excellent quality work (ex, found the most bugs during a bug bash), engineers who are power users of our quality services (ex, first engineer to utilize our library to mock customer data in a test), or engineers doing good quality work (ex: engineer that closed the most bugs and updated our dashboard). We then write up a blurb about what they did and why it's so impactful. Then, we post it on our company's kudos page. We also send the commendation to their managers and send them a custom Quality Champion T-Shirt. We find that if we incentivize and celebrate engineers who buy into our quality system, we will start to build champions for our program, which means more engineering interaction and value are created.

5. The Problem: Engineering Leadership is Not Investing in Quality Engineering Growth

Do your Quality Engineering programs sell your quality services to senior leadership (technical tooling, process, culture shifts, etc.)? Do they lay out a strategy on why growth in the Quality Engineering program is necessary to expand these services? Quality Programs generally have no problem following their charter and executing well, but are they utilizing the right muscles to justify growth? Does your Head of Engineering understand why they must continually invest in the Quality Program?

John Warrillow, author of The Automatic Customer: Creating a Subscription Business in Any Industry, writes, "*Your biggest competitor for your subscription business is not the rival service; it is your customer's*

inertia in not using your service. [6]" SaaS companies also struggle with getting their current customers to understand why they should not just continually use their products but upgrade and expand in utilizing more services (and generate more profit!) Like SaaS companies, our quality programs also need some sort of sales motion to advocate for themselves continually.

5.1 Selling Quality Engineering as a Service

So, how do we ensure our leadership team is bought in on our growth strategy? Here are two ways I utilize my sales pitch: to explain why they should be bought into our quality services and why these services need to grow.

5.1.2 Quality Engineering Service Delivery Review (SDR)

Once a quarter, we have a high-level quality service delivery review with our engineering leadership team and product and customer support leadership. During this review, I reviewed the data similarly to our QA reports but overlayed the current in-flight projects and their impact on this data. This is how I show the value of what we are doing. Then, I showcase areas where we are doing poorly, the effect if we don't alleviate these, and what initiatives the QA Program can execute to make an impact. I present our upcoming roadmap, the deliverables we will be doing, and what the outcomes will be. Finally, I always treat the last portion as a sales pitch to the leadership organization. I tier the results, showing what we can accomplish with the current resources, what we can achieve if we add resources and scale, and the impact of each decision.

5.1.3 Engagement with Leadership Organization

As the leader of my Quality Program, I utilize repetition to help sell its value. I aim to meet with each member of the senior engineering leadership to understand what areas they are concerned about, present them with data (QA Report, QA SDR), and discuss how the Quality Organization can help them. This relationship is essential to cultivate to ensure I align with the quality program, which is working and giving value to my leaders. This also allows me to understand how each leader plans to scale their team and will enable me to pitch them the utilization of more of our Quality Services. Doing this helps secure internal champions for the Quality Program and its growth.

6. Conclusion

The role of Quality Programs must evolve outside of just driving quality to reframe itself as a broader, value-driven service. By mirroring some of the operational strategies of SaaS companies, our Quality Programs will be successful in highlighting how indispensable they are in delivering business value. By understanding our internal customer needs, robust quality product management, proactive customer support, and effectively communicating success and areas for improvement to senior leadership, Quality Programs can secure their relevance within engineering organizations today.

References

1. Harry, B. (2017, June 28th) How we approach testing VSTS to enable continuous delivery. Retrieved from: https://devblogs.microsoft.com/bharry/testing-in-a-cloud-delivery-cadence/
2. Eland, M. (2019, Nov 29th) Partnering with Quality Assurance. Retrieved from: https://killalldefects.com/2019/11/29/partnering-with-quality-assurance/

3. Raileanu, G. (2016, Aug 18th) The Leading Reasons for Customer Churn in SaaS. Retrieved from: https://www.slideshare.net/slideshow/the-leading-reasons-for-customer-churn-in-saas/65129018
4. Caudill, D. (2023, Dec 3rd) Maybe Getting Rid of Your QA Team was Bad, Actually. Retrieved from: https://davidkcaudill.medium.com/maybe-getting-rid-of-your-qa-team-was-bad-actually-52c408bd048b
5. Gartner. (n.d.) How Service Leaders Can Increase Customer Loyalty. Retrieved from: https://www.gartner.com/en/customer-service-support/insights/customer-loyalty
6. Warrillow, John. The Automatic Customer: Creating a Subscription Business in Any Industry, 2015.