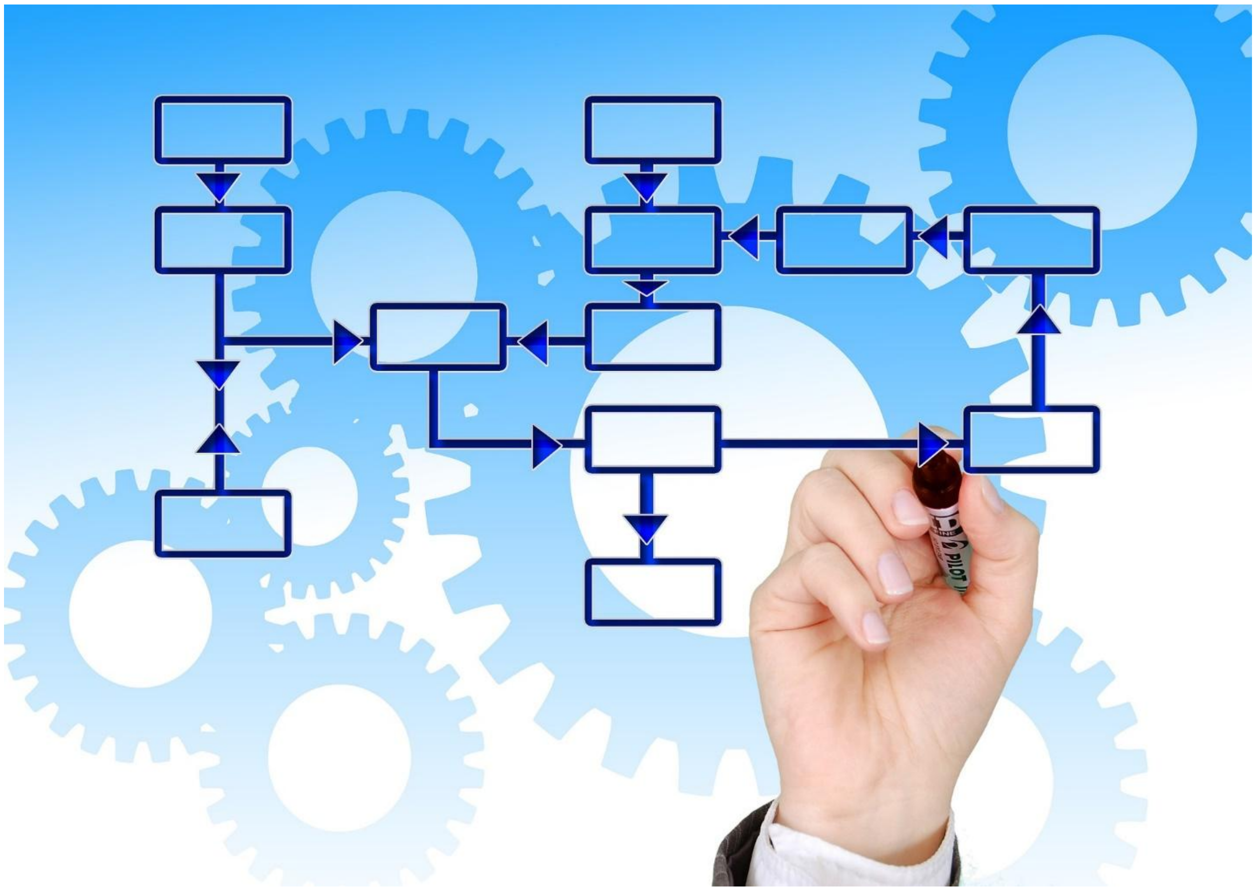
Yanking the root: process and root cause analysis



First, a quick recap

In a recent blog, the [Top 10 benefits of using on-shore testers](), one of the benefits mentioned is that on-shore testing provides increased efficiency. Onshore testers like Lighthouse's certified testers are quick to identify inefficiencies and then make corrections and improvements as they go. In the Lean Six Sigma world, we call these opportunities "quick wins," where you can identify a problem and correct it right away. Having a team with the skills, experience, and initiative to do that is worth its weight in gold, and these quick wins benefit your organization beyond that single project with improved efficiencies in your process.

It's not always that easy though

The entire reason Lean Six Sigma exists as a problem-solving methodology is that problems aren't resolved as easily as a quick win. Tell me if you've heard this one:

"Something's wrong, but I don't know why it keeps occurring, or how to really fix it."

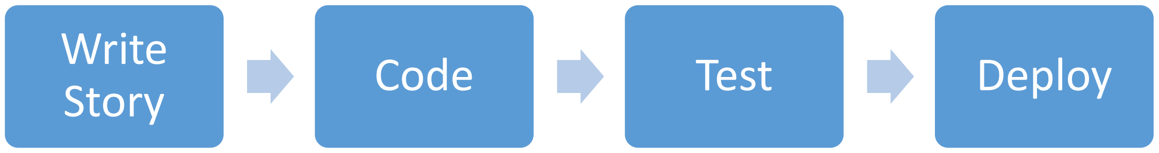
If you are nodding your head, you aren't alone. We've all been somewhere in the middle of that sentence at work or home because EVERYTHING we do involves a process. Let's take a few minutes today to dive into how to identify a problem and increase efficiency at the same time with a simple process documentation exercise.

Don’t go at it alone - involve the team

As you are mapping out a process, have people with you that are involved with that process as active participants. They are your best resources to uncover every detail, as well as pointing out where problems and bottlenecks occur based on their experiences.

Map out the process

Let’s assume your software development lifecycle (SDLC) looked like this.



Break out and expand the detail of the process

For some reason, you keep arriving to your release date late and riddled with defects. To dig deeper, we want to break down the process so we can see more detail. When we do this, focus on a few key areas:

- Understand where hand-offs happen between team members and why each hand-off happens.
- Understand the process from the perspective of the deliverable (the code) as well as from the perspective of each team member
- Add a typical duration to each step in the process to help prioritize where to focus your investigation

Identify variances and bottlenecks

Once mapped in detail, what we want to do is identify variances and bottlenecks. **Variances** are steps in the process that we have little or no control over. **Bottlenecks** are places where the process slows down, and can also be the result of a variance.

Start by reviewing each step of the process and think of what can go wrong. As you do, think of quick wins that could permanently change the process. Also, identify alternatives to fix an issue in the event of a variance (sometimes known as a workaround). Having workarounds identified upfront can help you modify your process quickly when it counts most. You may also find after identifying variances and bottlenecks that you need to break out your process to another level of granularity, or you may be able to talk through the issues you find as a result of identifying variances and bottlenecks as you see below.

1. **Issue**: the requirements didn’t include enough (or any) success criteria resulting in a variance because there wasn’t a common and full understanding of how the system should work.
    - **Solution**: you can prevent that by adding acceptance criteria to the definition of ready (DOR) and mandate that both positive and negative scenarios be included in the story or requirement.
    - **Also**, make sure testers have a chance to review stories ahead of developers building the code so they can ask the questions and confirm they understand how the system should and shouldn’t function. (quick win and prevents false positives)

2. **Issue**: testers waiting for something to test in the first week of the sprint while developers don't have time to fix all the bugs found at the end of the sprint. DING DING DING! We have many variables that can occur here:
    - Improperly sized stories (variance)
    - Scrum master not assigning some small stories to begin the sprint (variance/bottleneck)
    - Testers aren't reviewing upcoming stories and writing test plans for future sprints at the beginning of the sprint. (Have them do this and be productive – workaround)

Sometimes a bottleneck is the result of variance as you see here, while sometimes a step in the process IS the bottleneck, such as the underused or poorly used resource (way too typical) example.

Identify the value of each task

Value streaming is a step in process analysis to analyze each step for a specific value. This analysis is a critical step in identifying sources of waste in the process. In a value stream analysis, we identify each step as:

Value-add

The customer or stakeholder values this task and is willing to pay for it.

Business-value-add

There is no value to the customer, but it is a business requirement. One example may revolve around law and regulations.

Non-value-add

There is no value to the business or the customer. Explore ideas to eliminate this task from the process.

Translation back to our example

Your end-user is willing to pay for you to be on time and defect-free. To accomplish that, being prepared, efficient, and on the same page as a team are all steps of the process that are important to your end-user. These are called **value-add** steps.

The steps in the process that are important to you and your internal team, but not your end-user are the **business value-add** steps.

- Arriving on time to work is an arguable step, but an alternative could be to effectively use billable hours. There may be modifications possible, but these steps are still deemed necessary in the process.

The five whys

As you analyze process steps and need to dig deeper, the **five whys** technique is a great tool to do exactly that. Ask 'why' action is necessary, and keep asking 'why' until there is no longer an answer. You can usually get to your bottom answer within five iterations, but ask as many times as you need to.

Example

A JROTC drill team was conducting a drill exercise. Part of the drill had one cadet take four steps to the left, then three steps backward. It looked nothing short of awkward, so a mom inquired about the purpose of that part of the drill.

Why? (#1 to the cadets): that's how the drill captain taught us

Why? (#2 to the drill captain): that's how the seniors taught me last year, Master Sargeant would know.

Why? (#3 to Master Sargeant): that's the way we've always done it – *A-ha!*

Why? (start digging in history books): In the original drill (during the Civil War), the soldier took four side-steps and stepped back to make way for the horse pulling the cannon.

There was no horse and no cannon on the gymnasium floor. No wonder the whole thing looked so awkward! This step is unnecessary for quite some time since the military stopped using cannons and horses, so it is now a source of waste.

The same method of five why's can be used as a means of ferreting out defects, too. Just think how many awkward steps you could be taking in your software development for similar reasons.

Digging out the root cause

As you are identifying waste and bottlenecks, keep a critical eye on the specific steps where a problem occurs. Measurement data is likely to get you close if not spot on to the exact step and root cause, and your data will show if your cause is a **common cause** (recurring variance) or a **special cause** (rare unexpected variance). Common causes are fixable and worth the effort, so focus on those causes.

Lighthouse can get you heading True North

When it comes to software testing, processes are not carbon-copies of each other across industries, but there are some best-practices. Lighthouse offers a True North Testing Assessment that specializes in assessing the efficiency and effectiveness of your current software testing process. Using measurements to qualify the findings, expert assessors to identify what you are doing well, and what gaps suggest a course correction to avoid the 3 Perils of Software Development (defects, delays, and dollars).

Contact us for a risk-free assessment, and save your organization countless man-hours and hundreds of thousands of dollar in defects and waste.