# Sustaining in an Agile World

**Don Hanson II**

don@black-box.com

## Abstract

The challenges associated with sustaining a successful product often take managers by surprise.

Most sustaining planning discussions follow along the lines of “that would be a wonderful problem to have, as it means we will have acquired paying customers! Bah! We’ll cross that bridge when we get there.”

When you acquire paying customers and are faced with the problem of resolving customer escalations in a timely fashion, the reactive nature of sustaining a product presents challenges with agile development methodologies. Over time these issues can chip away at a team’s velocity, morale and ultimately retention.

There are two common approaches for handling sustaining; whole team sustaining and dedicated sustaining teams. Each has strengths and weaknesses that must be actively managed and overcome.

This paper introduces a new, blended approach that provides many of the positives and minimizes the negatives of the previous two. We’ll discuss the warning signs of the two common approaches to help you decide if this new approach is right for you. Then we’ll cover the five simple best practices for easily apply this new approach with your team.

## Biography

Don Hanson is bringing game changing new security management and protection capabilities to market. He is the engineering leader for the Data Exchange Layer, the foundation of Intel’s Security Connected vision, and Threat Intelligence Exchange open ecosystem.

Don Hanson founded his first company and began writing commercial software in the early 90's developing an evolving line of animation plug-in products. He has since worked on projects ranging from the mobile client for a wireless navigation startup to enterprise-level commercial software products. Don later spent several years delivering the next generation of McAfee’s flagship enterprise security management product and taking it to the cloud.

# 1 Introduction

Sustaining is the process of handling customer escalations involving your software product. The process may involve multiple levels of customer support personal attempting to troubleshoot the issue, or it could be a “Contact Support” form in your app that sends an email to a common account. In this paper we’re going to look at the sustaining process from the product development team leader’s perspective.

When a customer escalation reaches the product development team, whatever people, process, or sacrificial chickens you’ve put in place to handle dissatisfied customers have been exhausted. It is now up to you and your team to solve the problem posthaste.

When considering whom to assign to fix a particular escalation, there are two common approaches.

The first approach involve could be summarized as “you broke it, you fix it”. In this approach team members building the new, un-released version of the product are also assigned to fix escalations in the “old” version.

The second approach is on the opposite end of the spectrum in terms of who does the work to address escalations. This approach could be described as having a “dedicated fix-it team”, where specific team members work exclusively on addressing customer escalations. These team members do not work on the new, un-released version of the product.

Each approach has non-trivial advantages and disadvantages.

In this paper we’re going to discuss a third option that we call “rotating roles”. This approach combines aspects of the previous two to reduce the negative costs to the team while delivering outstanding customer support for escalations.

# 2 Welcome to Product Sustaining

The good news is congratulations are in order! You’ve created a product, released it and someone, somewhere is using it.

The bad news is something doesn’t appear to be working for them as expected. They would really appreciate it if you could drop everything you’re doing and make it work for them, say in the next five minutes or so?

Oh, and the clock is ticking. Whether the metric is a formal Service Level Agreement (SLA), lost revenue or simply customer angst, the sooner you can fix their issue the better.

# 3 Whole Team Sustaining – “You broke it, you fix it”

Whether your process is waterfall, agile or something in between, a common reaction is to assign the issue to the team member with the most experience in the general area associated with the customer escalation.

This is can often be the team member who originally wrote the suspect code.

With person most familiar with troublesome code working on the issue, the fix is most likely to be done in the best possible manner for the product. E.g. strengthening the architecture instead of piling on another band aid.

On the downside, the interrupt drive nature of customer support runs counter to the agile development best practices of keeping sprints atomic.

In addition, there are many situations that increase the mental cost of task switching between working on sprint stories and addressing sustaining issues beyond that of "simply" switching between stories. For example, the new version of the product may use 3rd party libraries incompatible with the released version, requiring separate development environments. Or the test automation may have changed between the two versions.

It can be hard to sustain forward progress on new features when team members who working on critical functionality, that is needed by other team members, are taken offline to address customer escalations.

This can lead to frustration among team members. It's easy to see how those blocked due to missing dependencies can be frustrated, but those pulled to address escalations often feel equally frustrated that they're letting the team down by not finishing their stories.

This can lead to team strife as frustrations mount. Equally bad is the potential for low quality code to sneak into your product as team members try to do more than they should, in order to keep up the appearance of meeting the sprint goals *and* the sustaining SLA.


**Figure 1 Summary of whole team sustaining approach benefits and challenges**

# 4 Dedicated Sustaining Team

Another popular approach for handling product sustaining duties is to create a dedicated sustaining team. Sustaining team members work only on customer escalations. Typically they are rewarded for meeting the sustaining SLA, which is addressing customer escalations within specified time limits.

This approach generally starts off doing quite well. However, the drawbacks build up over time.

Three common challenges to this approach are decreasing knowledge of product & architecture, divergent rewards and staffing the sustaining team.

Dedicated sustaining teams are typically created in part or whole from mainline developers with current knowledge of the product and its architecture. Over time the sustaining team's knowledge of the product

and architecture becomes outdated, as new features are created, technologies updated, design patterns changed by the mainline team.

This can result in sustaining fixes that use libraries the mainline team is trying to move away from, duplicate functionality available elsewhere in the code base, or negatively affect product performance or usability in another area.

Hearing about mainline team members backing out sustaining changes is a telltale sign.

Over time the divergent rewards for the sustaining and mainline product development teams tend to drive their cultures apart.

The mainline product development team is rewarded for making a product that provides useful new functionality to customers and is engaging to use. The sustaining team is rewarded for making customer complaints go away.

The problem is that it is often quickest and easiest to make customer complaints go away in ways that run counter to improving the product functionality and ease of use.

For example, it may be easier and quicker to write a monitoring program that restarts a web service when it crashes, than to eliminate the crash itself. This example is a bit extreme, but I've seen it used.

The third common challenge is keeping the sustaining team adequately staffed. Sustaining is not typically seen as an attractive role. It can be challenging to staff a dedicated sustaining team with appropriately skilled and talented team members.

Tactics to address this include using the sustaining team as a training ground for junior developers, or requiring all new team members to spend time on the sustaining team before joining the mainline team.

Unfortunately these tactics virtually guarantee team members unfamiliar with the product and its architecture and/or having entry level skills will be addressing customer escalations. Expect escalations to be fixed with band aids, instead of root level, structural changes.

Note that each of the above is a general trend. The quality, productivity and morale impacts typically build up under the radar. They may be first detected as a niggling sense of something being not quite right, or in the unexpected departure of a lead developer. There are exceptions, but managing these challenges over time is time consuming and hard to do well.

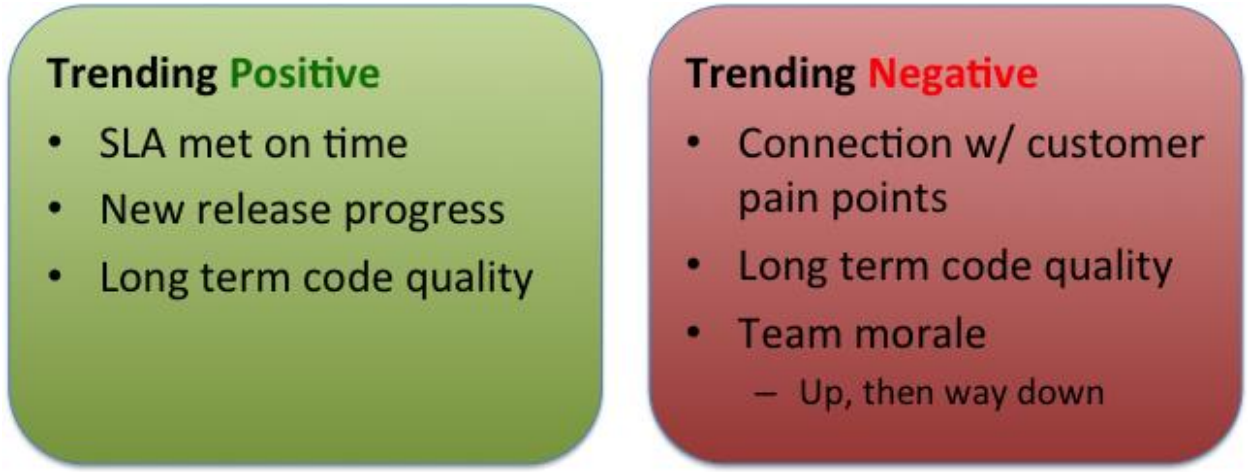


**Figure 2 Summary of dedicated sustaining team approach benefits and challenges**

# 5 Why Settle for Trade-offs?

While either approach may solve a number of problems in the short run, they can be difficult to maintain productively over time. At the daily team level, terrific work using team dashboards, Kanban and other systems to raise the visibility of the sustaining tasks and SLA metrics is being done throughout our industry.

As an agile advocate, at the program design level both approaches fall well short of what should be possible.

The challenge is we want the best from each approach. We want to meet the SLA and maintain new feature delivery velocity. We want team members to be aware of and feel responsible for the pain points encountered by customers. We want the sustaining team members to be familiar with where the next release is headed and how the issues should be fixed to maintain long term code quality. And we want engaged team members who believe their sustaining efforts are a valuable use of their time.

# 6 Rotating Roles Pattern

The rotating roles pattern combines aspects of the previous approaches to reduce the costs to the mainline effort while delivering outstanding customer support for escalations in the best possible manner for the product.

As an added bonus it can increase team morale by leveraging the optimism & energy from team members newly rotating into role to address structural and non-trivial process issues.

## 6.1 Concept

The rotating roles pattern is incredibly simple; team members take on a particular role for a period of time. Generally everyone on the team take a turn in the role.

You may already be practicing the rotating roles pattern. For example, many teams see value in having everyone take a turn as build master.

## 6.2 Best Practices

There are five best practices required to make this work optimally for your sustaining effort.

First, enough sustaining roles are needed to cover all aspects of the sustaining process. That may be a single person, one person to triage and one to fix issues, or a web developer and a database guru, whatever is appropriate for your team and situation.

Every team must decide where to draw the line for sustaining roles. For example, many teams do not create an installer sustaining role. They create a hotfix and/or patch installer template, and borrow time from the installer guru if needed. Some teams have their rotating sustaining roles handle hot fixes and creating patches, but involve the mainline team for validating patches (in those rare cases when the automation doesn't cover everything!).

Second, the sustaining role is 100%. Sustaining team members do not work on mainline stories. On larger teams they may have their own stand-ups in lieu of mainline stand-ups.

This helps achieve three things; it prevents conflict of interest decisions between sustaining and mainline development responsibilities, keeps the set of sustaining roles as few in number as possible, and

provides potential time for sustaining team members to use their own judgment to “do good things” related to sustaining the product if there are no open escalations.

Third, sustaining team members are 100% responsible for the team meeting their SLA commitments. There are no fingers to point, no excuses to give, responsibility for meeting the SLA stops here. That’s a lot of responsibility, so we give sustaining team members one special power to help meet the SLA goals

Fourth, sustaining team members can ask for help, on an interrupt basis, from any non-sustaining team member at any time.

But wait! Why do we bother with dedicated sustaining roles if they can interrupt the mainline team, you ask?

This is the magic sauce that makes it all work in the real world. In the real world team members are not fungible; product teams have a mix of senior and junior developers, newbies and old hands, specialists in different product areas or technologies. At pre-set levels before an escalation goes out of SLA compliance, such as 60% and 90%, sustaining team members are advised to get additional help from the mainline team.

In most cases group dynamics work to motivate sustaining team members to do everything in their power to avoid requesting help. The team lead, architect, scrum master or manager can help subtly emphasize this by having out-going sustaining team members demonstrate what they fixed in the product, learned about customers or improved process-wise.

Fifth and finally, the length of service for the sustaining roles must be set. It is critical that the role lasts long enough for a sense of ownership to be established during each cycle, so think in terms of multiple sprints. We also want each sustaining team member to fully address far more escalations than they hand off to the next cycle’s incoming sustaining team members.

Depending on the average length of time required to address an escalation, this could be one month, two months or even a quarter.

## 6.3 Benefits

Over time many different, fresh perspectives are applied the process. Items that merely annoy one team member may be an irresistible thorn for another that must be addressed so peace, harmony and productivity are restored.

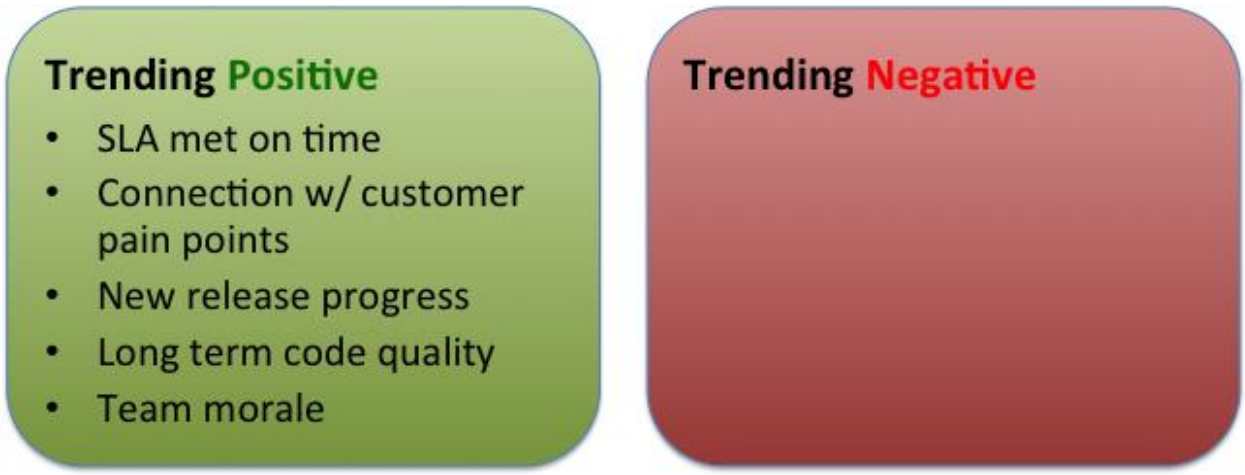


**Figure 3 Summary of rotating roles sustaining approach benefits and challenges**

# 7 Conclusion

Leveraging the rotating role pattern for your product sustaining efforts can help maintain team moral and enthusiasm while delivering new features and addressing customer escalations. Five essential best practices can help turn a serious challenge to team stability into a source of pride.

1. Create a set of dedicated, rotating roles to handle all aspects of the sustaining process.
2. Sustaining roles are 100% of assigned team member capacity.
3. Sustaining team members are 100% responsible for meeting the SLA
4. Sustaining team members can ask for help, on an interrupt basis, from anyone on the mainline team.
5. The sustaining role rotates on a longer timeframe, such as monthly or quarterly.

Thank you for your time. For questions, comments or insights I can be reached at don@black-box.com.