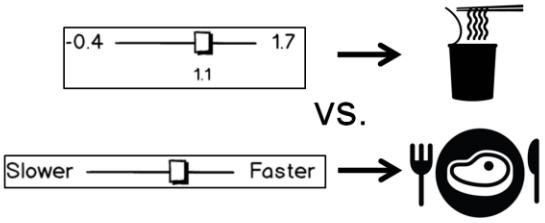
Sustaining in an Agile World

Don Hanson
Director of Engineering, Intel Security

Welcome! My name is Don Hanson and I'm here to talk about some of the challenges, and options, for Sustaining a product in an Agile World.

Who Am I?

- 22 years in software industry
- Started animation plug-in company
- Very early focus on User Experience
- Love to tinker
  - Currently own 4 3D printers, (3 of which work!)
  - Drone enthusiast
  - Nomad desktop CNC gathering dust ☹



- Worked primarily on high risk, high reward, software projects
  - Most recently, Data Exchange Layer – a communication fabric that is the foundation of Intel Security's Security Connected vision.

2

So who am I? I've been in the industry for some time. I learned the value of a good User Experience at my first company.
Where it meant the difference between Ramen one month and steak the next.
- I love to tinker. 4 3D printers. (3 working) Drone enthusiast.
- Kickstarter junkie
I've been fortunate to work primarily on high risk, high reward projects. Everything from green field to massive next-gen re-architectures.
Most recently working on the Data Exchange Layer, a communication fabric that is the foundation of Intel Security's Security Connected vision.

Who Is This Presentation *Not* For?

People working on…

- Cloud based project with automatic feature gating*

Feature gating:

- Code check-ins are fully tested and released to successively larger groups of customers via automation w/ no human interaction.
- This is how new Facebook developers are able to commit code into the production environment on their first day.

*I would like to talk to you afterwards.

3

So who is the presentation not for?

Who Has Shipped Commercial Software?

Raise your hand if you've released software that is sold to customers.

Congratulations!

4

Who Has Sustained Commercial Software?

Keep your hand up if you've dealt with customer escalations on software you shipped.

All right! You may have experienced some of the things we're going to talk about.

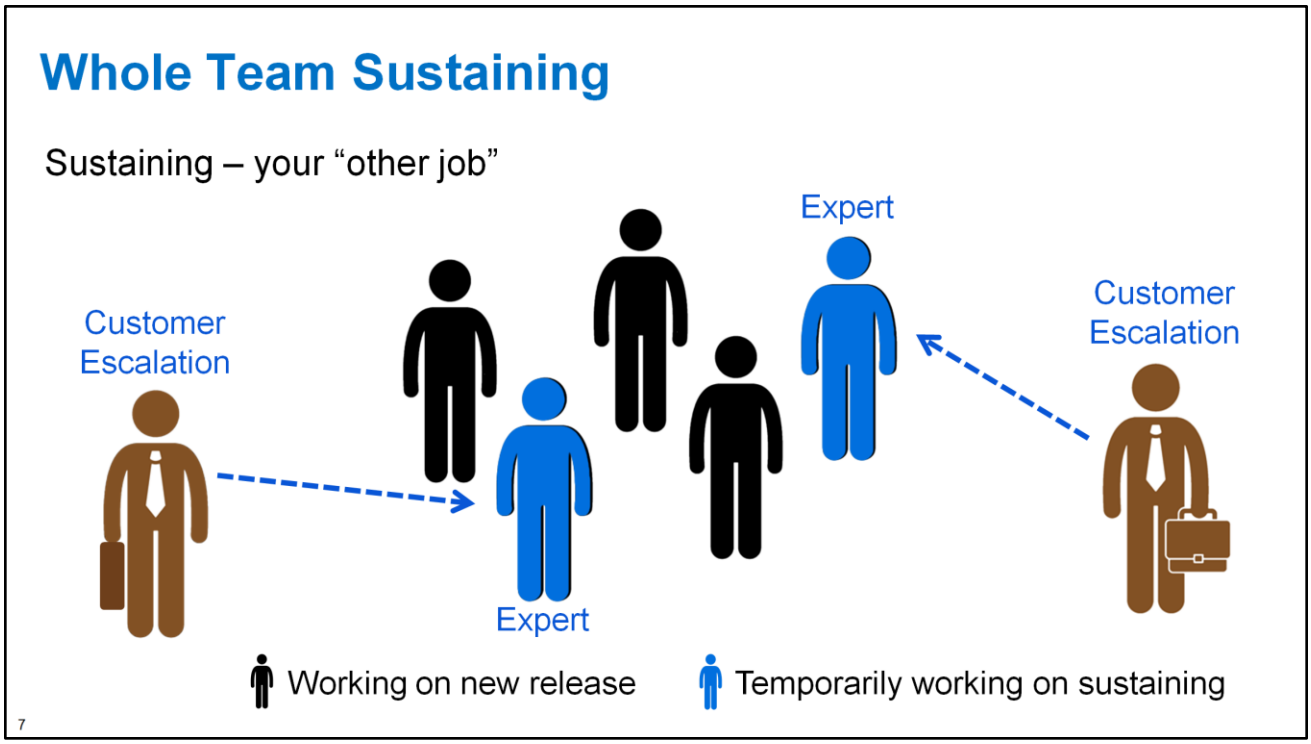
5

Definition of Sustaining

- Responding to customer complaints about software product
- Performed by development team
  - Typically support or helpdesk team was unable to resolve
  - Issue has been escalated to the development team, "Customer Escalation"
- Resolution is often a code change
  - Hot fix or patch
- Performance indicator on how quickly customer escalations are resolved
  - The dreaded "Service Level Agreement" (SLA)

6

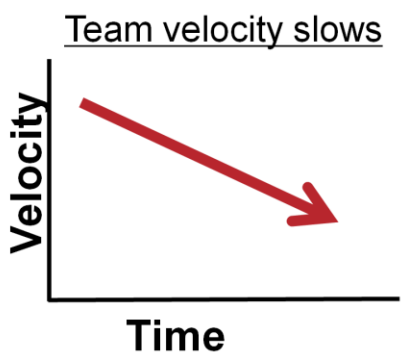
Product sustaining is a broad topic. For our purposes, I am referring to:
1) Responding to Customer complaints
2) Performed by Development team
 - (your support/helpdesk unable to solve, "escalated" issue to development)
3) Resolution is often a code change
 - (typically hot fix or patch)
4) Some form of a Performance metric that tracks how quickly issues are closed. (also known as the dreaded SLA)

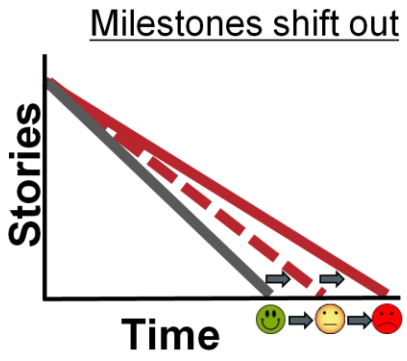


Many groups start with the "Whole Team Sustaining" method.
You've released the product.
Customers are buying it.
The team is happily working on the next version.
...
As you can imagine this has a bit of an impact...

Warning Signs Appear

- Team members blocked from completing stories or tasks.
- Customer escalation resolutions slow or incomplete.





8

We start seeing Team members blocked
Escalations take longer to fix or are incomplete.
Ultimately
- Velocity slows on next release.
- Predicted milestones shift out.

What's Going On?

Breaking a number of Agile best practices.

- Sprints are not atomic b/c scope changes due to escalations.
- Dependencies can't be managed.
- No accountability to mainline project.
- No "Agile knobs" to turn

Perhaps experiment with non-Agile options

- Reduce capacity for every team member?
  - *It burns us!*
  - Most of team does not get an escalation. More time needed from those who do.

9

So what's going on?
It turns out, we're Breaking number of Agile best practices.
- Escalations are interrupt driven, this causes Sprints not be atomic because the workload can change at any time.
- Dependencies can't be managed
- No accountability
- No "Agile knobs"

Brilliant Idea!

We'll use that *(somewhat tongue-in-cheek)* Agile Blocking pattern!

- "The needs of the many outweigh the needs of the few, or the one."

And lo, the Dedicated Sustaining team was born!

10

You think about it and come up with a brilliant idea
Leverage that Agile Blocking pattern
Needs of the many outweigh the needs of the few

And lo,
the Dedicated Sustaining team was born

Equilibrium Restored!

- Customer escalations are resolved within the SLA
- Mainline team velocity increases over time
- *There was much rejoicing…*

For a while…

11

You find Equilibrium has been restored!
Escalations are resolved quickly
Mainline Velocity increases
- there was much rejoicing ...

for a while...

New Problems Arise

Dedicated Sustaining Team Challenges

- Much slower closures on complex issues.
- Fixes start to address symptom, not the cause.
  - Over time sustaining team's knowledge of product & architecture stagnates.
- Staffing challenges

12

Over time, New problems arise
- complex issues take much longer to fix
- fixes start to address symptom, not the cause.

You realize the sustaining team's knowledge of the product & architecture becomes stale.

They lose awareness of how issues SHOULD be fixed, rather than COULD be fixed.

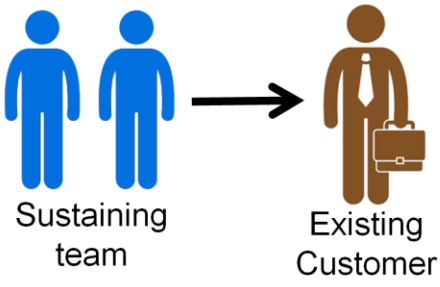
Staffing challenges on sustaining team.

Different Goals

Two Teams with Different Goals

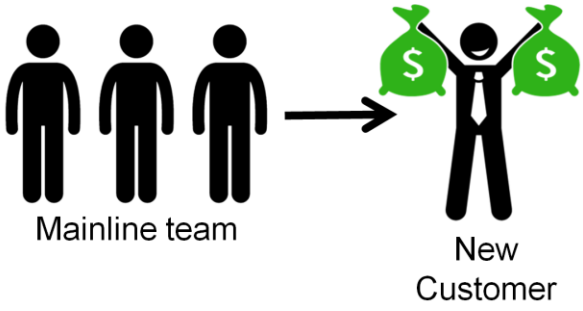
Sustaining Team

Make angry customers happy



Mainline Team

Attract new customers

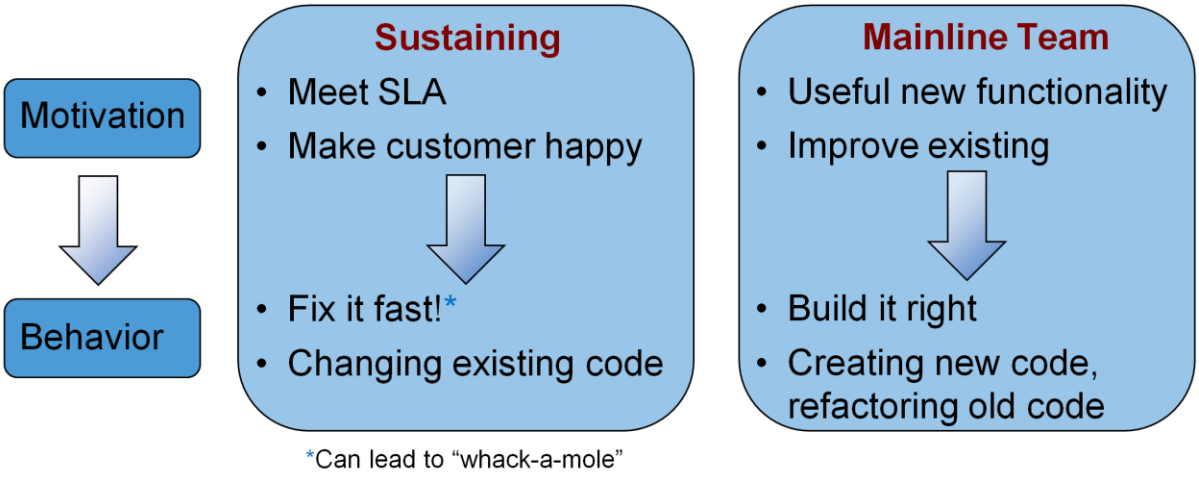


13

These issues stem from having two teams with different goals.
Sustaining -> make existing, angry customers happy
Mainline -> delivering useful new functionality & attract new customers

Behavior Follows Motivation

Different motivation drives different behavior.



14

Since Behavior tends to follow Motivation, the teams are pushed in very different directions.
Sustaining
Meet SLA, make customer happy -> Fix it fast!*
Performing mainly small changes to existing code.
*can lead to "whack-a-mole"

Mainline
Useful new functionality -> Build it right Primarily working on new code, or broad refactors of old code.
Do we have any other options?

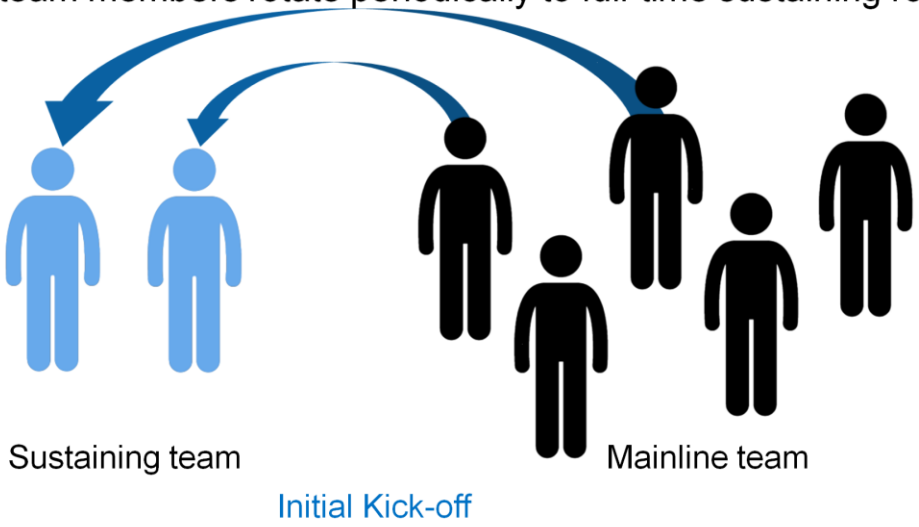
Happily we do.
There is another option to try.

It's a simple concept called Rotating Roles.

You may already be doing this in other areas,
such as buildmaster, which many teams hand off on a monthly or quarterly basis.

Rotating Roles

Mainline team members rotate periodically to full-time sustaining role.



16

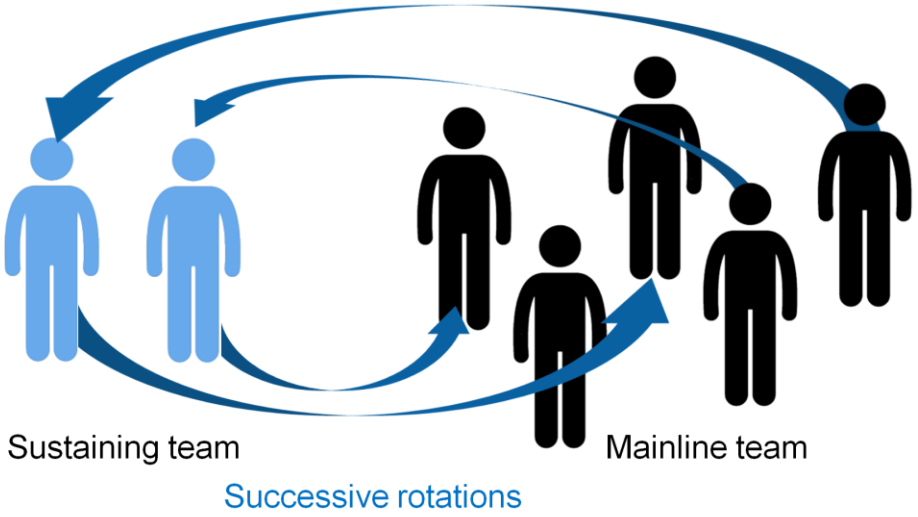
It's a simple concept called Rotating Roles.

You may already be doing this in other areas,
such as buildmaster, which many teams hand off on a monthly or quarterly basis.

With rotating roles, mainline "volunteers" form the sustaining team for a cycle.

Rotating Roles

Mainline team members rotate periodically to full-time sustaining role.



17

At the end of the cycle, they return to the mainline team
And new volunteers are selected.

Five Best Practices

Sustaining Team Rotating Roles Best Practices

1. Sustaining team must be large enough to succeed on it's own.
2. Sustaining role is 100%.
   - Sustaining team never works on mainline stories.
3. Sustaining team owns meeting SLA.
   - Everyone must be impacted if SLA not met.
4. Can request mainline assistance at any time on interrupt basis.
5. Rotation cycle must be long enough to fully address escalations
   - Typically multiple sprints.

18

To make this work, there are 5 critical best practices.
Similar to Agile, these practices are self-reinforcing.
1. team Large enough
2. role is 100%. Avoid conflict of interest.
3. Sustaining team owns meeting SLA
   - but Everyone must be impacted if not met.
4. In this case, with great responsibility comes a little bit of power.
   - Request assistance on interrupt basis at any time from anyone
5. Long rotation cycles

Benefits

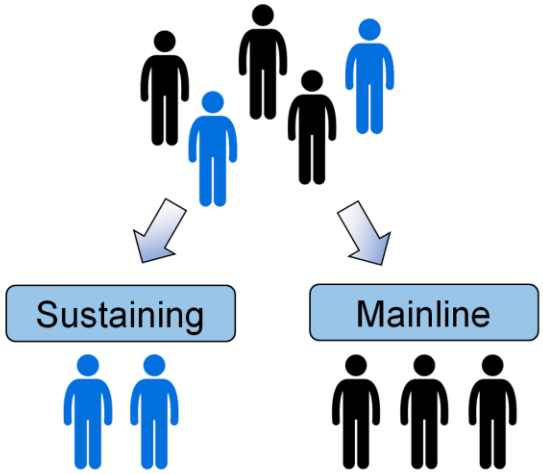
Benefits of dedicated, rotating roles.

Sustaining Team

- Escalations quickly resolved
- Fixes done in best possible manner
- Knowledge of long-term direction of architecture & code base

Mainline Team

- Predictable sprints
- Increased velocity
- Appreciation of sustaining needs



19

This allows Team members focus on one role at at time, but understand needs of both.

Escalations are fixed in best possible manner, address root cause - not symptom.

Predictability returns to sprint progress,
velocity increases

Historical Evidence

Our experience with Rotating Roles turned out really well…

- Senior team member kicked off sustaining rotation.
  - Amazing what they can do when they focus on one thing.
- Predicted 3 quarters to clear backlog, did it in 1 quarter.
- 26 extremely hard-to-fix, out-of-compliance issues addressed.
  - One issue over 400 days old
  - Host of minor issues cleared out
- Over time incoming rate of escalations declined.
  - Fixing the cause, not the symptom, had noticeable effect.

20

Our experience with Rotating Roles for Sustaining has turned out really well.
On one project…
A Senior team member jumped in on first cycle.
Cleared out backlog that had built up over a few years in 1 quarter, instead of 3.
26 hard, out-of-compliance issues fixed
- 1 had been open over a year
- host of minor issues addressed
Ultimately, incoming rate of escalations declined as team fixed root causes, instead of symptoms.

Q&A

21

Thank you for your time.

Thank You

Don Hanson
don@black-box.com
https://www.linkedin.com/in/donhanson

22