

Skimming Content Only: Large Server Hardware Company (pseudo name)

James Carpenter



Audio versions of this case study are [available here](#)

Synopsis

I strongly believe the best chance of long-term success is achieved with a broad product boundary coupled with feature teams executing within the full scope of the product. If the art of the possible excludes this choice, an incremental approach starting with *extended* (more cross-functional) component teams and aggressively moving towards *expanded* (more cross-component) component teams and feature teams can still be worth pursuing.

Benefits of this strategy within Nakashima's Modular Compute System division included:

- Locally improved **adaptability** and **value delivery** within the extended component boundary
- Improved technical practices that created **improved quality**, along with improved awareness of what additional improvements could bring
- Early identification and resolution of **defects** related to the extended component
- Improved employee **collaboration, engagement, and learning** within the extended component teams
- Increased awareness of **organizational impediments** and the need to make even more organizational changes

I hope you will find our success inspirational and instructive. Similarly, I hope you will avoid repeating our mistakes and instead spend your energy learning as you make new ones.

Skimming Hints

The case study is extremely long and detailed. If you prefer to skim it very quickly I recommend reading just the following:

- Synopsis section
- All figures and associated captions
- Conclusion section

ALL OTHER CONTENT HAS BEEN DELETED FROM THIS SKIMMING COPY

Full Case Study Is Better

Far greater insight is available if you are willing to take the time to read the full case study. The full case study is approximately 74 pages in length.

<https://less.works/case-studies/large-server-hardware-company>

- This online English version has been translated by the LeSS community into Japanese and Chinese. Please see the upper right-hand corner of LeSS website to select your desired language.
- Printing of the online version seems to work much better in Safari than Chrome. See the link below for English versions explicitly formatted for printing.

https://agilecarpentry.com/case_study_recording/

- Audio versions in podcast format
- YouTube audio version with figures displayed as the video content
- Full case study manually formatted in both Letter and A4 format
- Amazon printed versions of full case study
- Abbreviated skimming version of the case study with synopsis, figures, and summary only; manually formatted in both Letter and A4 format

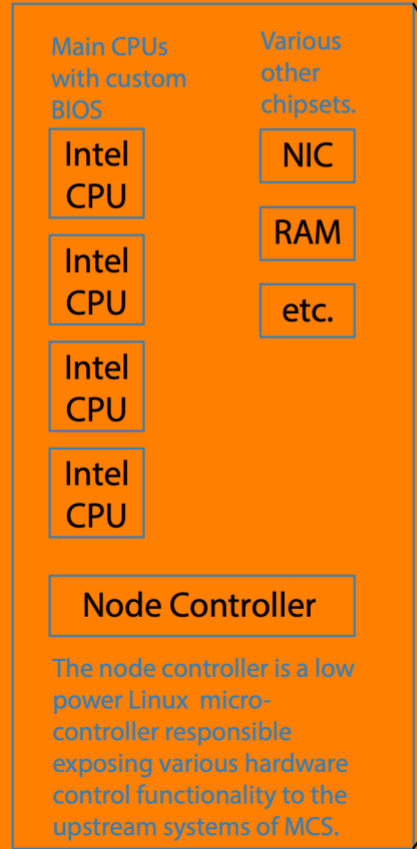
Actual MCS Product Boundary

Hardware Generations

prod-N ... prod-1 prod in-dev

Supported MCS hardware gen. is a negotiable component dimension

Blade Motherboard Detail



MCS Administrator

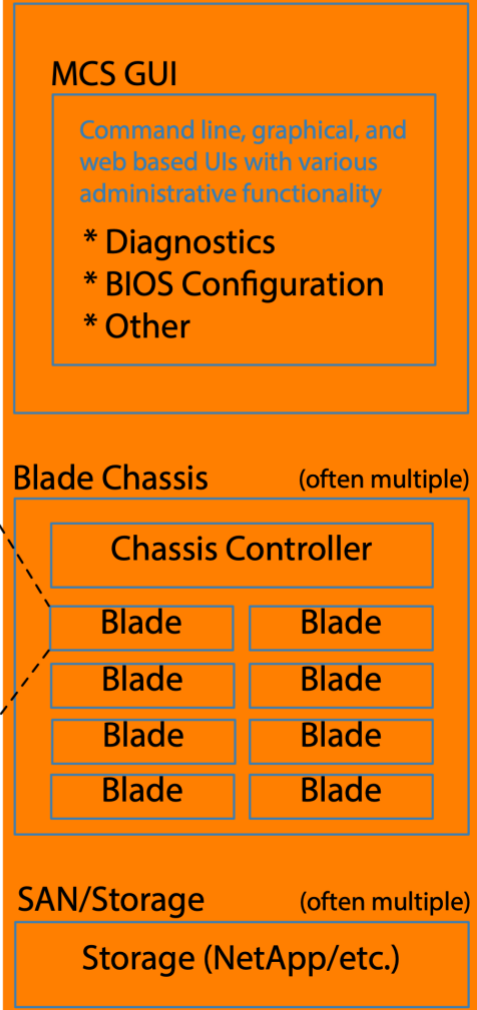


Figure 1: From the perspective of the Product Management group, Nakashima divisional boundaries, and external customers the natural product boundary includes the entire system of network, compute, and storage capability. Even broader product boundaries are possible but are not that practical as the coupling between other systems is sufficiently standardized to be interchangeable with data center hardware and software from a variety of vendors. These other systems do play a minor role in the larger scale testing scopes, but they are not the focus of Modular Compute System testing.

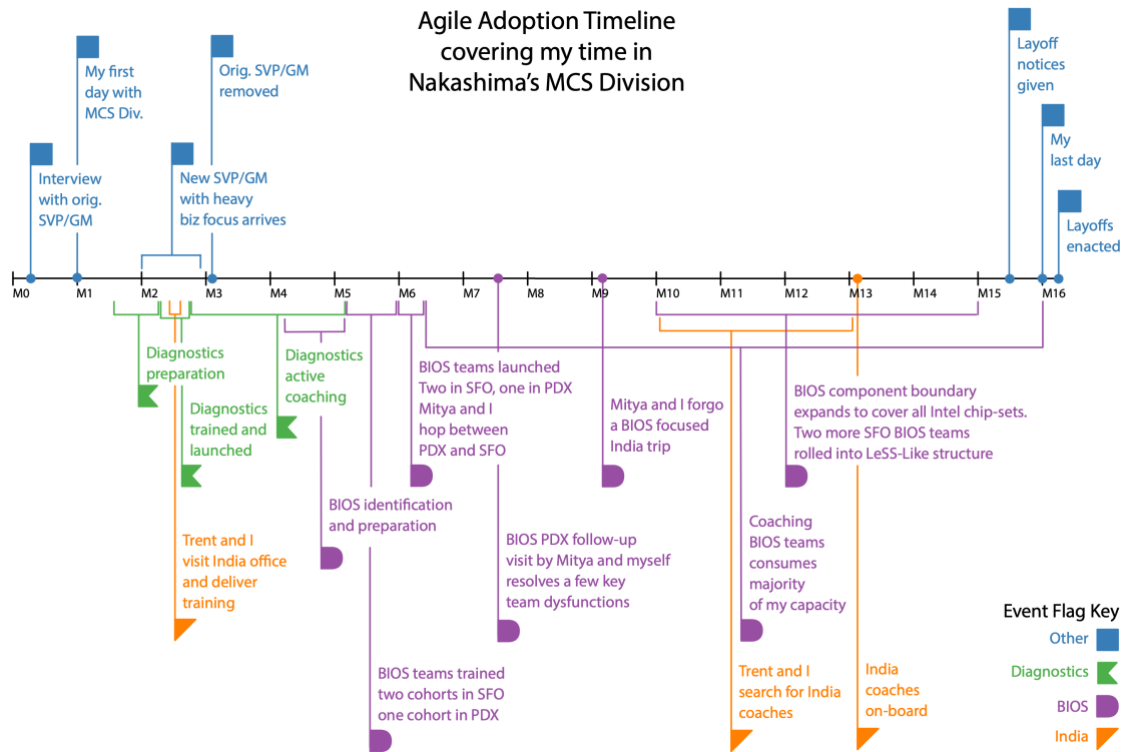


Figure 2: With the various overlapping adoption efforts and managerial changes it is easy to lose track of the overall story arch. Hopefully, this timeline will help you keep track.

Components Affected by Diagnostic Feature Set

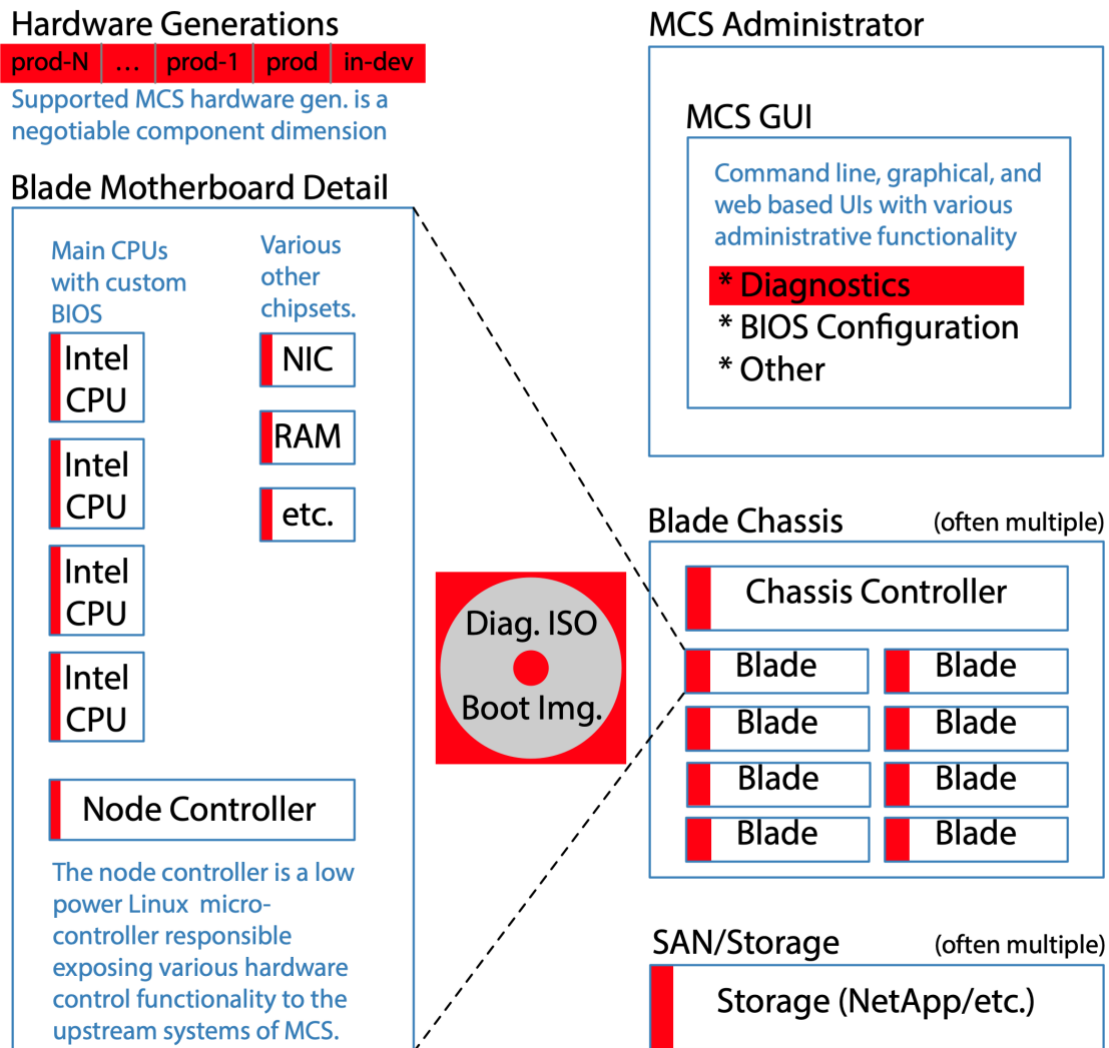


Figure 3: Although the diagnostic code is largely encapsulated in a custom ISO image and some diagnostic focused code in the MCSA, the diagnostics capabilities are broadly focused and relevant to all hardware generations. Detailed knowledge of each component in the entire MCS system is often required. When a given MCS component's firmware is missing the ability to probe it, the diagnostic team adds it.

Feature Team Adoption Map for Diagnostic Team

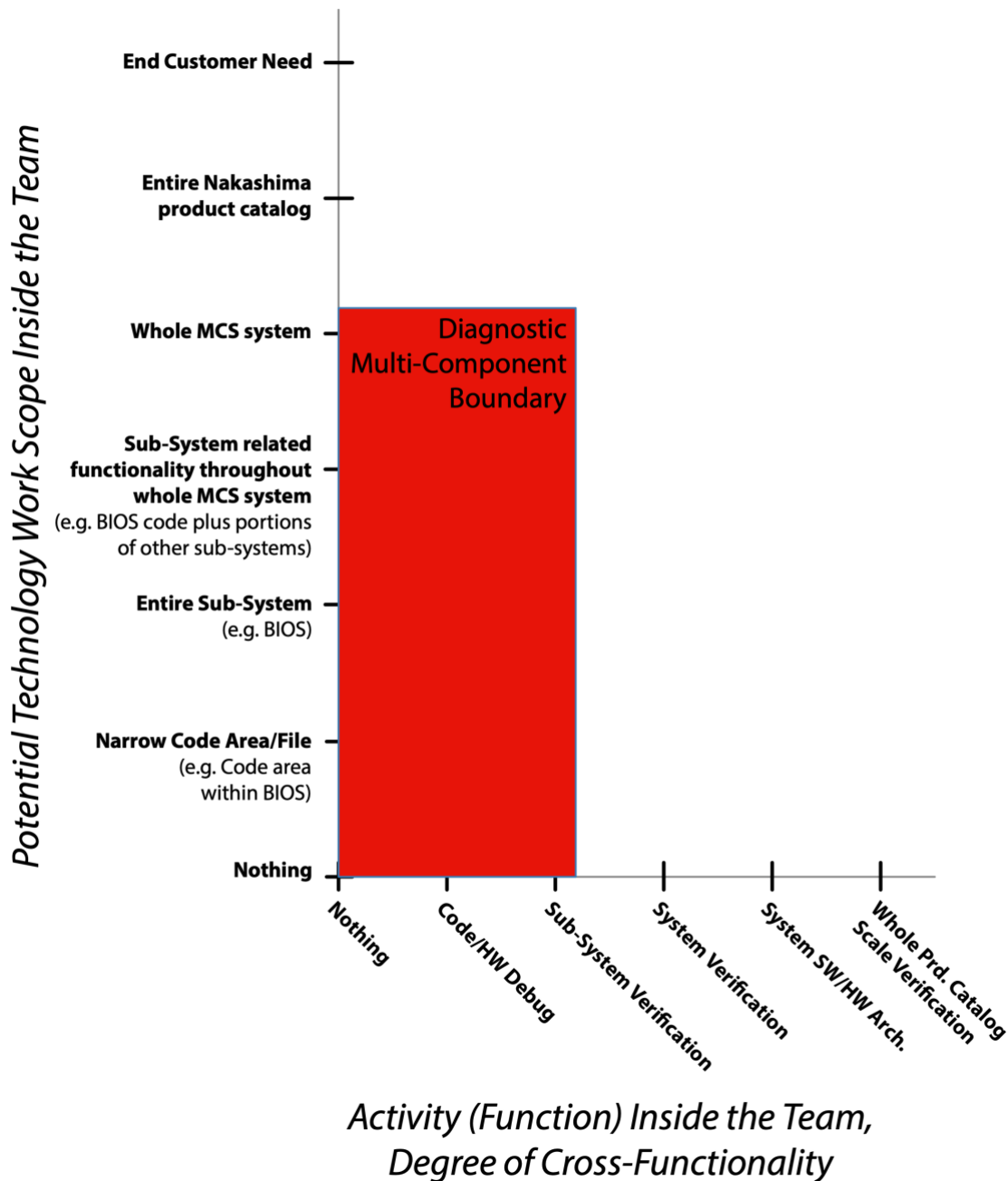


Figure 4: The diagnostic capability is narrowly focused on helping customers diagnose problems with their on-site MCS systems, particularly hardware component failures. Although the focus is narrow, the scope spans the whole MCS system. More information on the usage of Feature Team Adoption maps can be found at: https://less.works/less/adoption/feature-team-adoption_map



Figure 5: Here is a photo of the diagnostics Scrum development team. Pairing and swarming became more common over time, though full mob programming never quite caught on. The team had both the test and development talent needed to deliver a potentially shippable increment at the end of each Sprint. The development team used a physical task board. The meeting room we took over was a bit smaller than we would have liked.

Second Definition of Done (Firmware Feature Team in Waterfall Ecosystem)

Version: 2.03

Date: July 5, 2017

Common

- Zero bugs at the end of the Sprint within the Dev team's control to fix.
- Any similar functionality between the stand-alone and integrated solution has been factored into common shared libraries.
- Manually validated against each available hardware platform profile.
- Release notes updated and published to common location.
- Configuration information and Release note information saved in source control in a manner which makes it easy for the technical writing team to write documentation for the release.
- Peer Review completed. (Pair programming counts as peer review.)
- Demo to Product Owner

Specific to Stand-alone Solution

- Automated unit test created for both positive and negative cases for any new or modified functionality within custom code. Not required for third-party libraries developed outside the team.
- Manual Regression executed and passed for all platforms.
- Passed all solution level test cases for new defects (manual testing today).
- Final release image build by centralized build systems team is successful.
- Optimal image size should not increase beyond 100MB.

Specific to Larger Solution

- Code committed to main branch of overall solution.
 - Automated smoke tests extended to cover any new or modified functionality.
 - Existing commit tests pass.
 - No new static analysis errors introduced.
 - Manual feature test passed.
 - No relevant open defects.
 - No relevant open regression defects.
-

Figure 6: The Definition of Done used by the diagnostics team evolved to what you see here after a few Sprints. The stand-alone portion of the Increment was provided to the field service division by the end of each Sprint. Providing the MCS integrated diagnostics capabilities to the end customer required waiting for a release of the waterfall developed MCS product. This example Definition of Done along with additional context can be found in Table 9.2 of Forging Change.

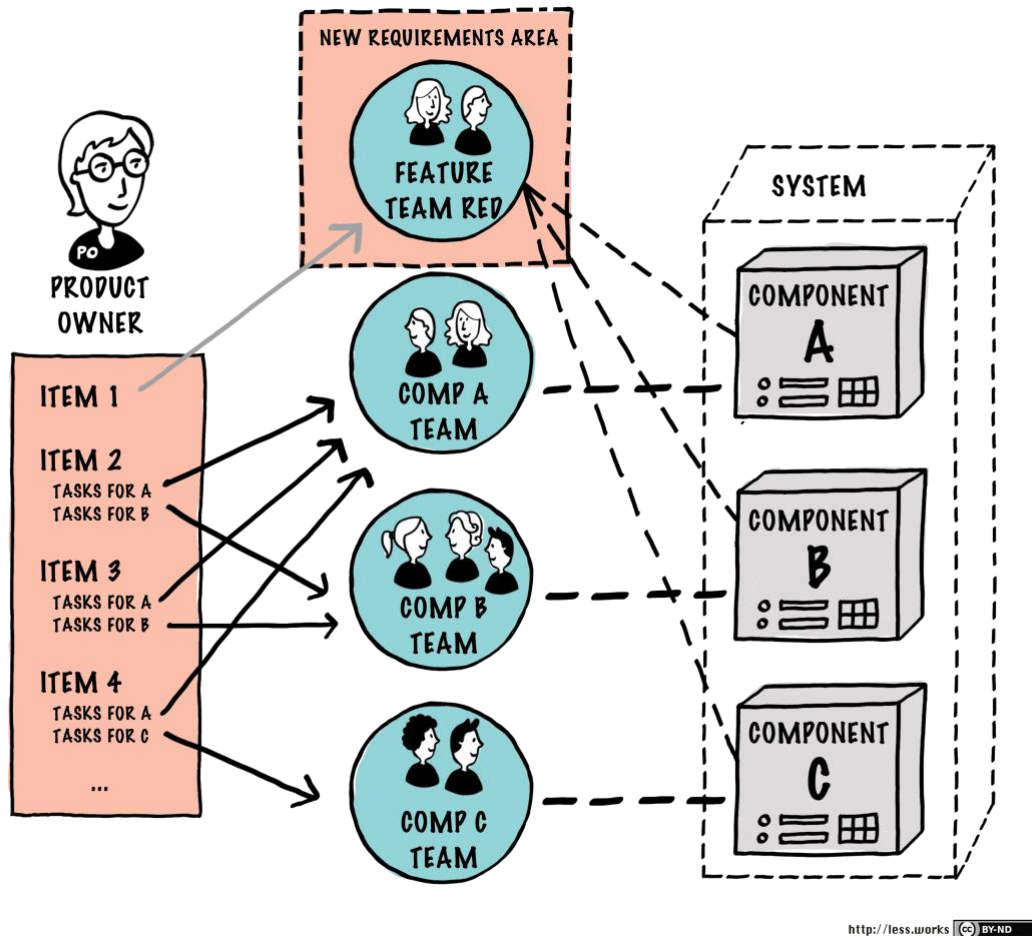


Figure 7: You will find this diagram as Figure 4.11 in *Large-Scale Scrum: More with LeSS* as part of the *Transitioning to Feature Teams* guide. You will notice Feature Team Red in its newly formed Requirement Area consumes from the same Product Backlog as do all the component teams. Although this loosely correlates to the diagnostic team situation, the day to day reality was slightly different. The diagnostic team was a real self-managing team free from the negative direct effects of the Contract Game. In contrast, most of the other MCS teams were subject to the Contract Game within a waterfall delivery context. The diagnostic team's organizational reporting relationships were never changed so as to intersect the organizational chart above the level at which the Contract Game was being played for the waterfall teams. This failure eventually resulted in the erosion of the supportive context required for the diagnostic team to remain successful. This failure became increasingly apparent after the departure of the original engineering SVP/GM.

Initial BIOS Component Boundary

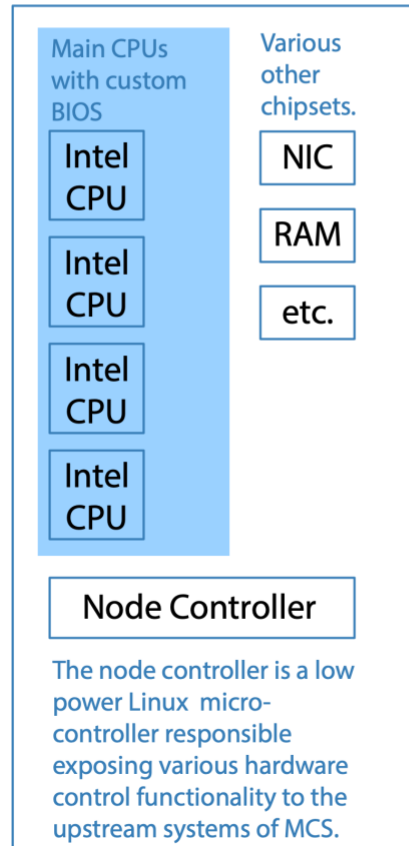
Hint: Compare with Expanded BIOS Multi-Component Boundary

Hardware Generations

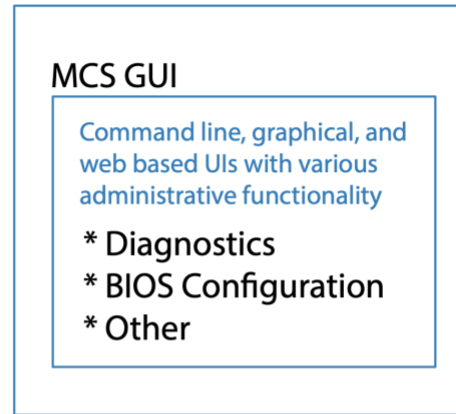
prod-N | ... | prod-1 | prod | **in-dev**

Supported MCS hardware gen. is a negotiable component dimension

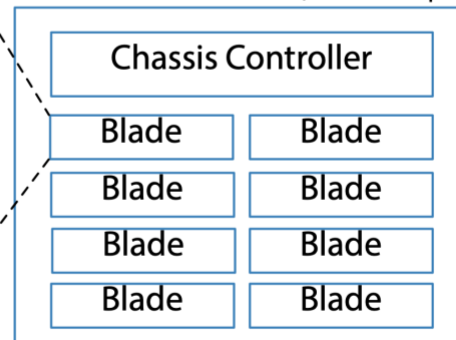
Blade Motherboard Detail



MCS Administrator



Blade Chassis (often multiple)



SAN/Storage (often multiple)

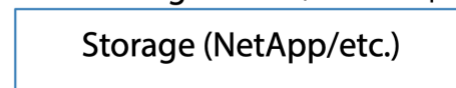


Figure 8: The initial BIOS component boundary only included the custom BIOS. Even at this narrower scope, it still included hundreds of specialized code areas within the custom BIOS itself and millions of lines of C code. Few if any of the several dozen BIOS engineers initially knew more than one or two aspects of the BIOS code.

Expanded BIOS Multi-Component Boundary

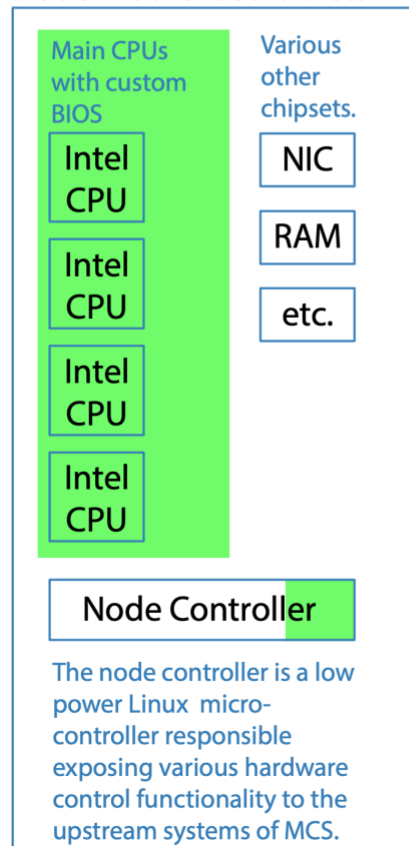
Hint: Compare with Initial BIOS Component Boundary

Hardware Generations

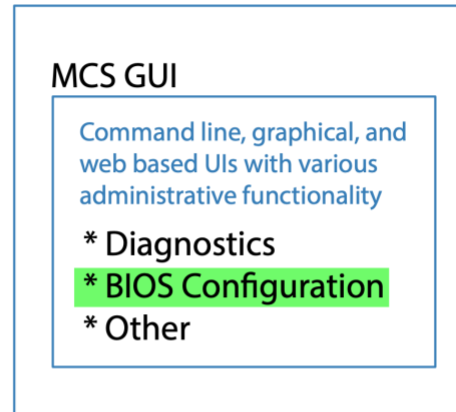
prod-N | ... | prod-1 | prod | **in-dev**

Supported MCS hardware gen. is a negotiable component dimension

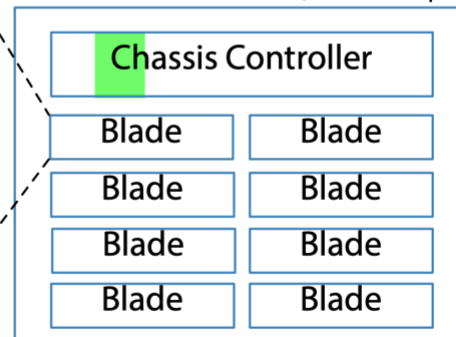
Blade Motherboard Detail



MCS Administrator



Blade Chassis (often multiple)



SAN/Storage (often multiple)

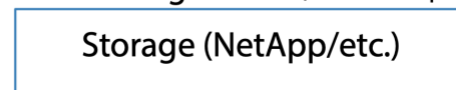


Figure 9: The expanded BIOS multi-component boundary included everything along the BIOS configuration control path. It mapped to a natural product requirement area, and could be easily understood by a Product Owner from the Product Management group. Although never fully realized, efforts to move in this direction were made.

Feature Team Adoption Map for BIOS Teams

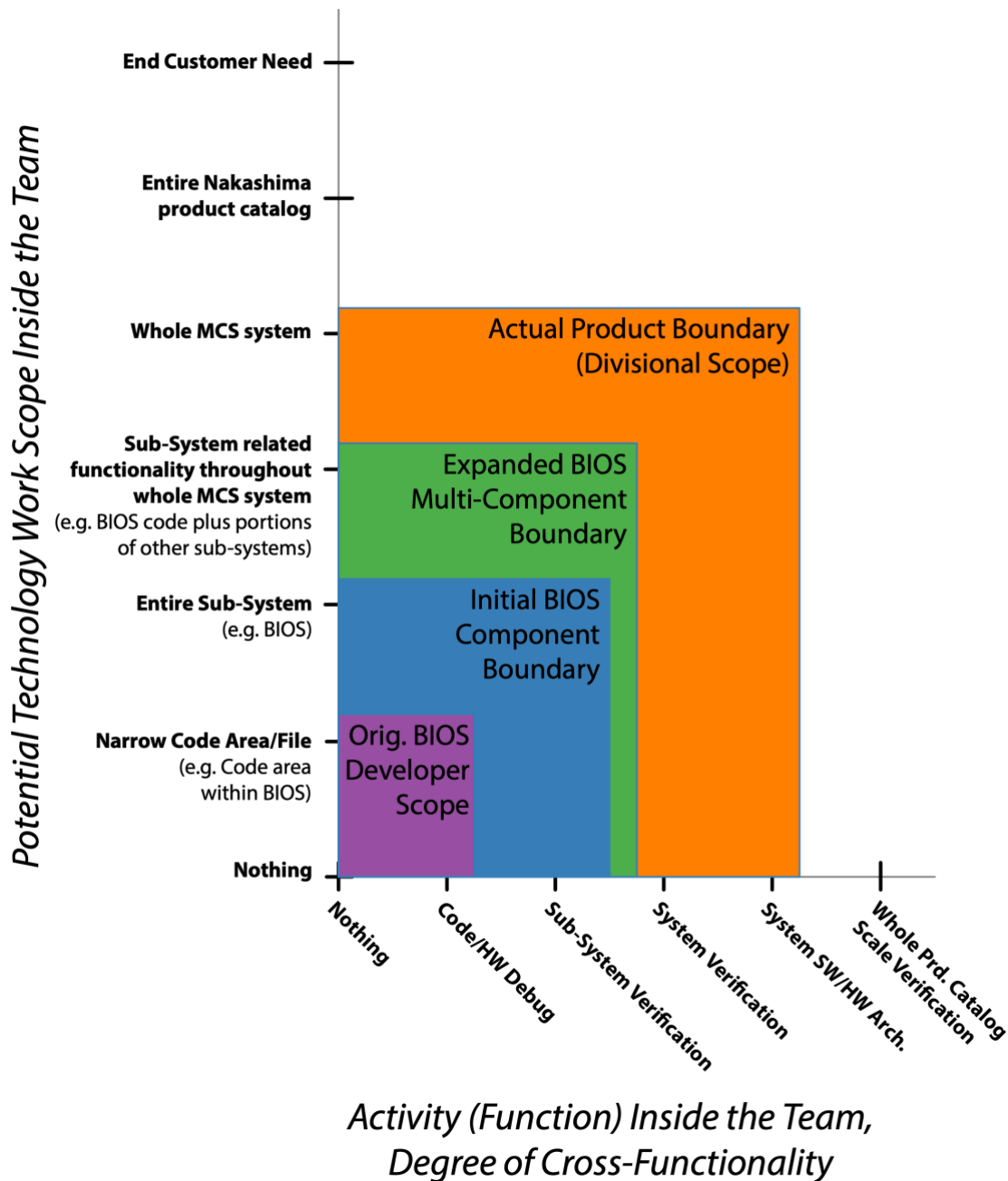


Figure 10: The BIOS developers were originally more of a loose group of a few dozen individuals who each specialized in a narrow aspect of the BIOS customization. The BIOS system alone contained millions of lines of code in an extremely esoteric system domain.

Original Organizational Structure with Original SVP

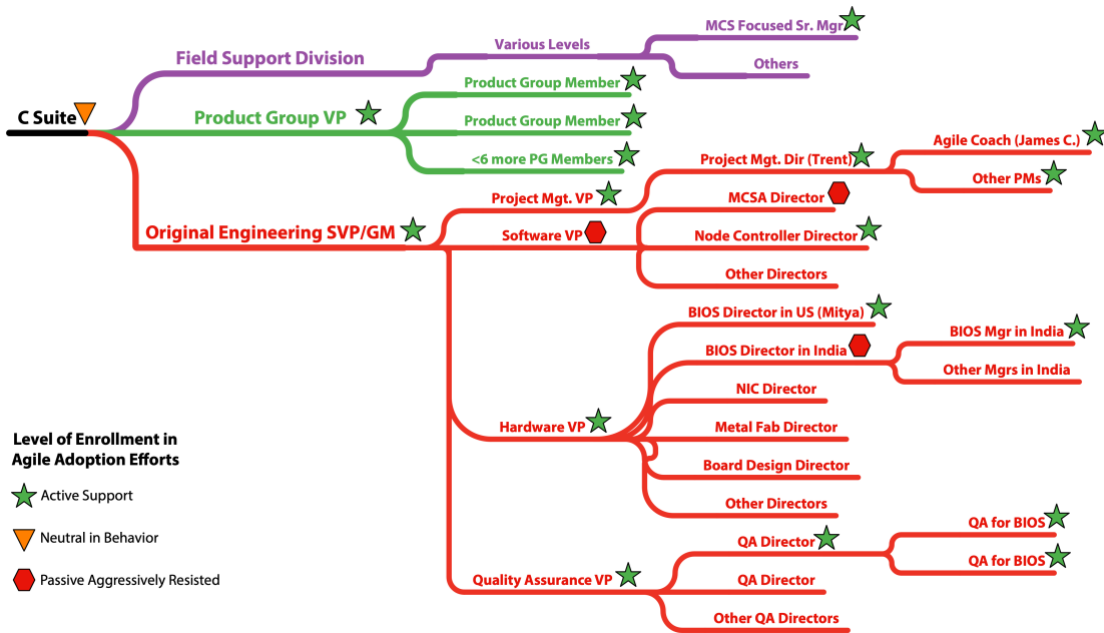


Figure 11: The initial Sr. VP/GM of engineering for the MCS division was extremely supportive of the agile adoption efforts. I also found active support throughout much of the organization. A large number of directors, managers, and individual contributors provided active guidance as I attempted to better understand and help the organization. Unfortunately, this Sr. VP's tenure was very short and there was a key VP responsible for the more pure software portions of the product who was passively aggressively opposed to any real change.

Organizational Structure After Early Change of Engineering SVP/GM

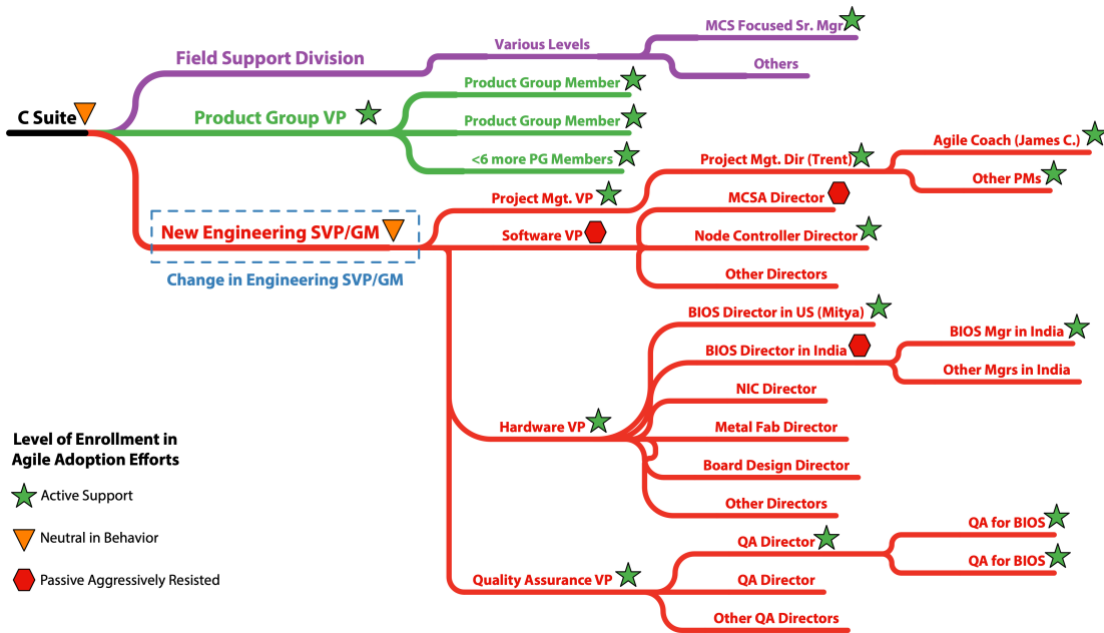


Figure 12: The initial Sr. VP/GM of engineering of the MCS Division only had his role for a few months before I arrived. Within a couple months of my arrival he was replaced with another Sr. VP. In retrospect it is obvious the new Sr. VP's direction from the C suite was to rationalize the size of the division. The new Sr. VP was almost completely unavailable to me and unwilling to actively engage in the agile transformation efforts. Although it generally happened outside of my view, I believe I continued to receive active support and air cover from the Project Management VP. Although there were some additional organizational changes over time once the new engineering SVP took over, none were very significant to the teams attempting an agile adoption until the Hardware VP left.

Organizational Structure After Departure of Hardware VP

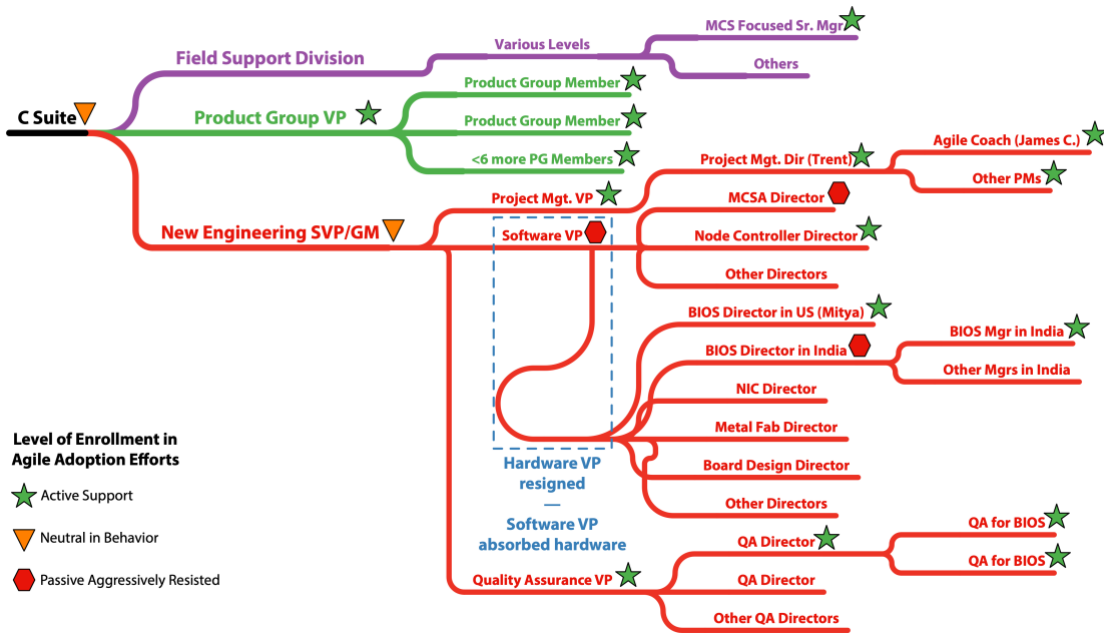


Figure 13: Under the cloud of upcoming and active layoffs many people began to depart the organization voluntarily. Around the same time a new extremely well funded startup began to actively recruit some of the more skillful engineers and managers in the MCS division. One of these departures was the Hardware VP who the BIOS teams had reported through. The new engineering SVP chose not to backfill the Hardware VP but instead to have all those previously reporting up through the Hardware VP report through the Software VP. As the Software VP was always passively aggressively working against the agile adoption efforts this did not bode well. Over the course of a few months half the BIOS team members were laid off, my engagement ended, and Mitya followed the Hardware VP to the same well funded startup the Hardware VP had left for. A little over a year later, Trent also left Nakashima Incorporated.

MCS Division People By Geography

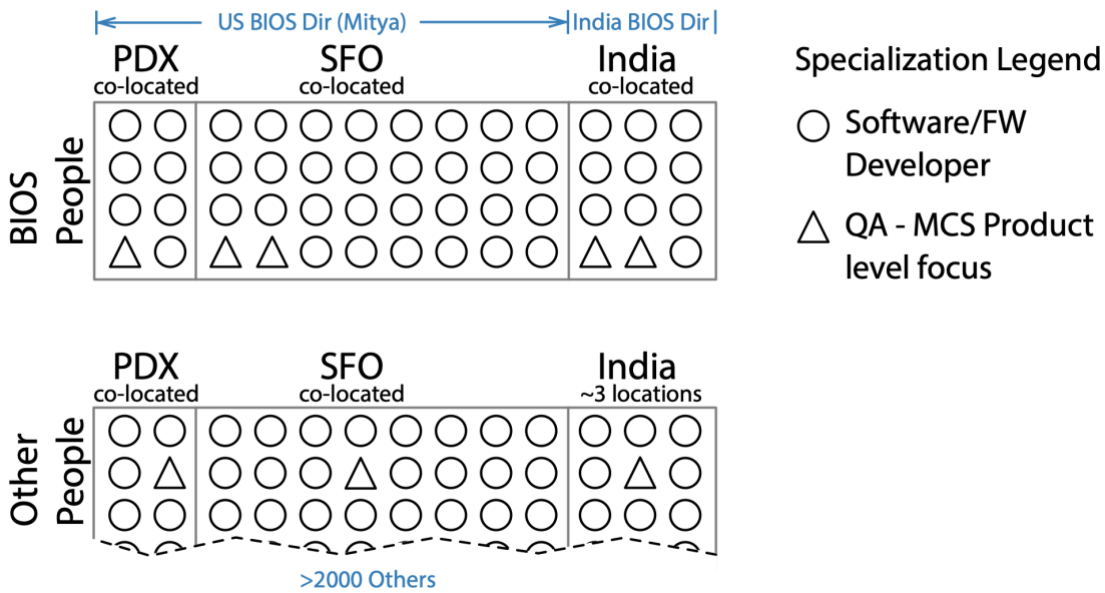
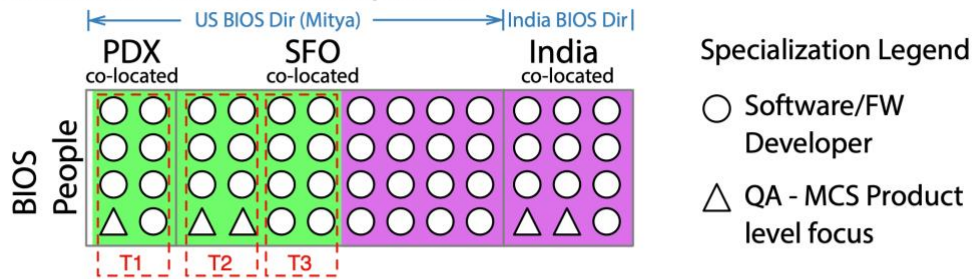


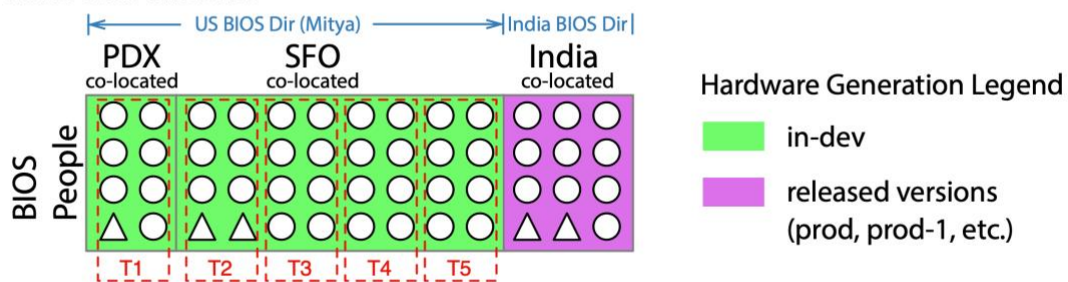
Figure 14: The vast majority of the people within the MCS Division are co-located on one or two floors within a single building in their respective cities. We were careful to ensure BIOS team members were generally sitting within a few feet of their teammates. Workstations were generally friendly to swarming. Half the work occurred in lab space so many team members effectively had two working locations. With the exception of a handful of the testing specialists, everyone in the US reported through Mitya. Even the testing specialists were co-located, fully allocated to BIOS, and treated just like everyone else on the teams. There were one or two BIOS developers who were dispersed but these were the only exceptions. Unfortunately, we didn't manage to officially remove specialist manager roles as part of a matrix structure. Nevertheless, managers didn't emphasize or hold onto specialism but supported people being multi-skilled.

BIOS Extended Component Team Expansion By MCS Hardware Generation

Initial BIOS extended component teams



After four months



After seven months

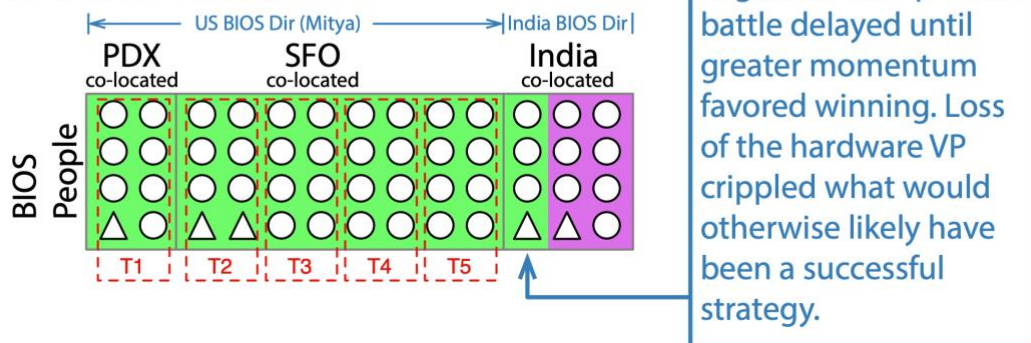


Figure 15: MCS hardware generation was used as one of the component dimensions in defining the boundaries of the LeSS-oriented adoption within the BIOS component. With the supportive hardware VP and a few trips to India it is likely Mitya and I would have been able to successfully work out the politics. Unfortunately, the change in the VP layer coupled with the layoffs precluded this strategy. The information in the timeline and organizational structure diagrams is relevant to what you see in this diagram.

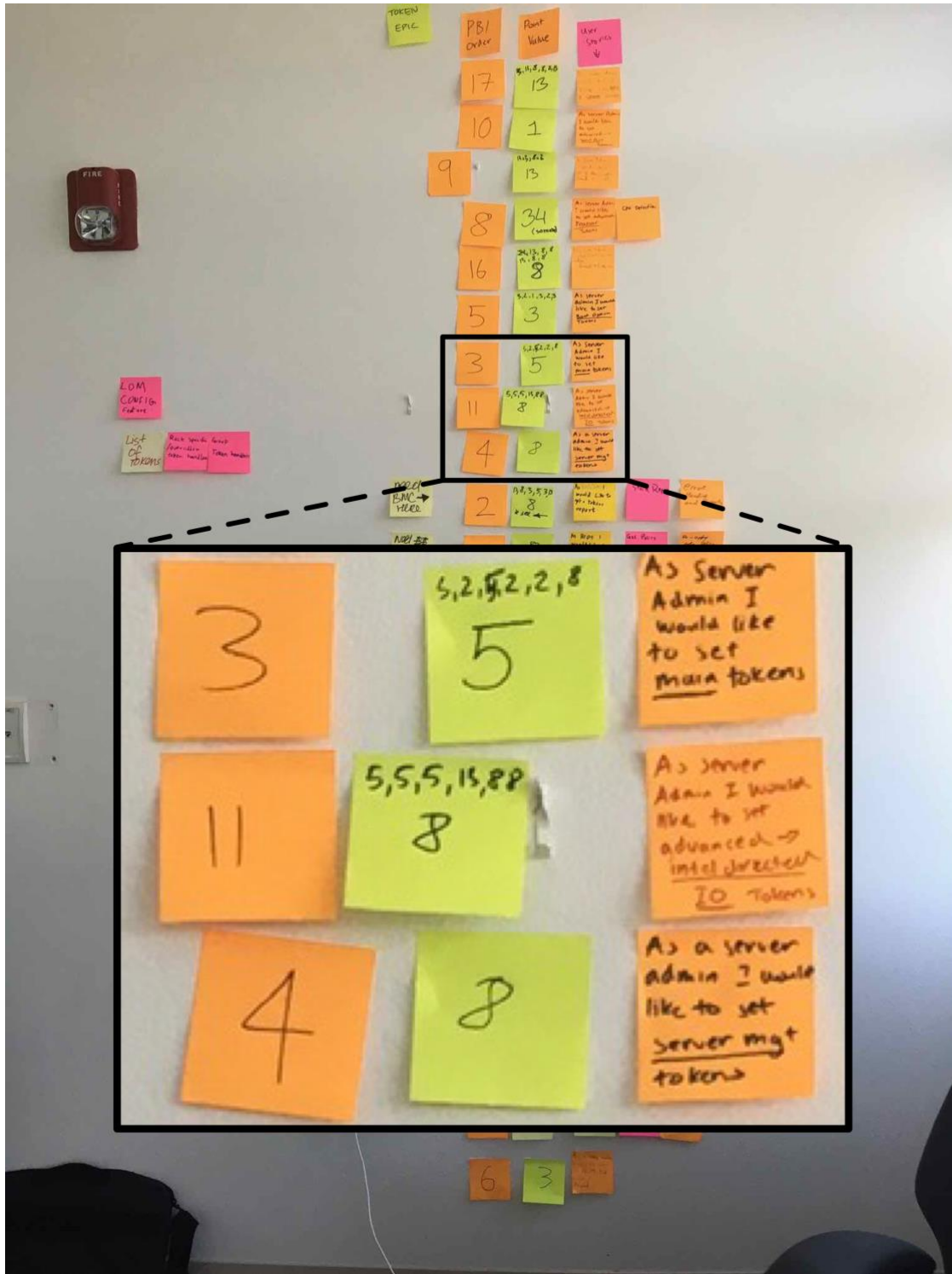


Figure 16: The initial BIOS component backlog brainstorming produced a series of preliminary PBI post-its with little more than short descriptions and/or titles. Each PBI post-it was assigned an effort estimate using poker planning, and given an appropriate order in the BIOS Component Backlog. Just before leaving we captured photos of our work in preparation for transitioning the data to electronic format.



Figure 17: The brief post-it note PBIs from the initial two day launch meeting were further refined and stored electronically. These refinement efforts were done by small cross-team groups focused on particular areas of the BIOS component. It took a Sprint or more before the cross-team groups reached a point of diminishing return. Now that we had enough additional insight from the cross-team refinement efforts; we returned to a physical format to help us see the bigger picture. Here you see the story cards printed out and randomly taped to the wall in preparation for more refinement activities.



Figure 18: The BIOS Fake Product Owner began to look for natural groupings and orderings. The end result was a bit of a mix between a story map and a snake-like ordered Component Backlog with epic groupings. This large map was slowly evolved over the course of several days. Various groupings of people from the BIOS feature teams would be pulled in for more insight as it made sense. As the wall settled down the Fake Product Owner made sure to call a meeting with every BIOS Scrum team member to conduct an overall sanity check. At this point the MVP had become evident as seen by the red arrow.

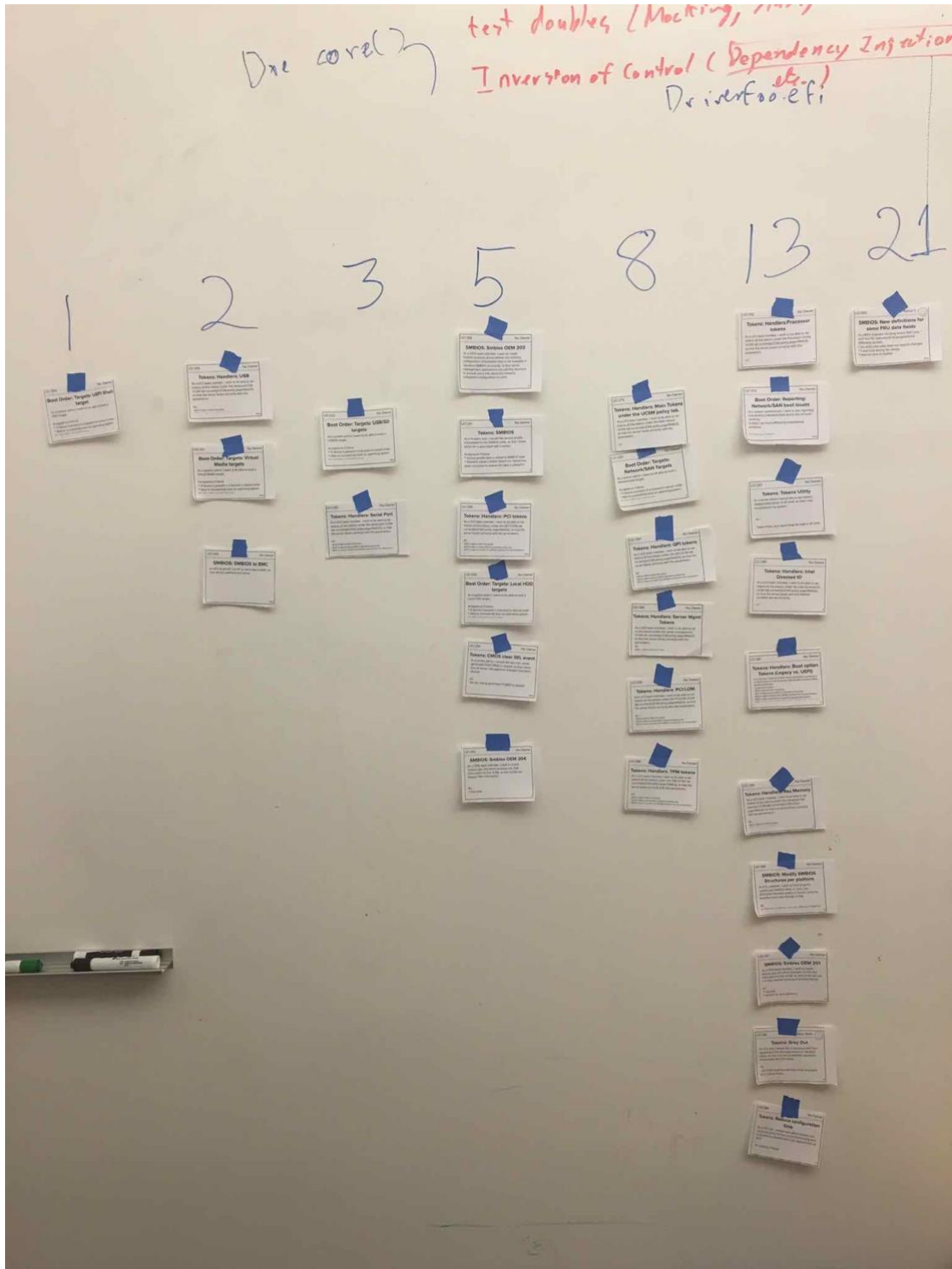


Figure 19: While every BIOS Scrum team member was available during the large group multi-team sanity check, affinity estimation was used to re-estimate every remaining PBI within the MVP. Afterwards the cards were rearranged into a cleaner story map version, and Rally was updated to reflect the new information.



Figure 20: Mitya ensured we had a helper to provide any masking tape we needed.

1. ALL below activities done w/ Latest bundle
2. Code Reviewed. No Core Change (unless explicitly agreed differently)
3. Code checked into official BRANCH
4. Documentation Complete
5. ALL test cases Executed
6. Sanity passes (BIOS,)
7. Bios Release Test Suite executed
8. Zero 1,2,3 Bugs
9. WORKS on all Platforms

ld infra (work in progress)

Figure 21: This is the initial draft Definition of Done created by the BIOS teams during their multi-day launch event.

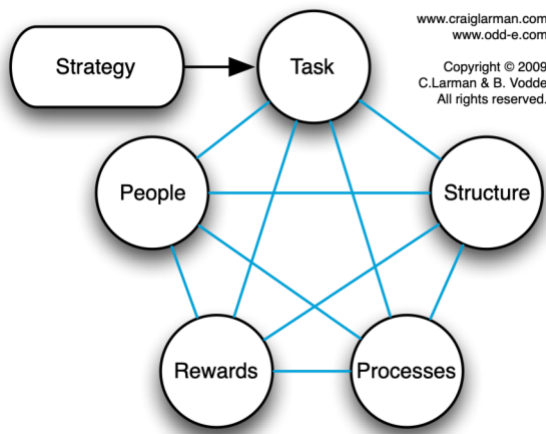
Third Definition of Done (Firmware Component Teams in Waterfall Ecosystem)

Version: 1.23

Date: July 3, 2017

- All User Story acceptance criteria have been met.
- Relevant Epic acceptance criteria have been met.
- All activities below performed with latest third-party release bundle.
- Code peer reviewed in electronic peer review system.
- No changes outside of pluggable layer, unless agreed by third-party as an exception not handled by pluggable layer.
- Code checked into official branch.
- Documentation complete.
- Features documented via SW Spec Template and checked into document management system.
- All test cases executed (manual testing allowed).
- Test plan reviewed.
- Test management system used to track manual test cases.
- Zero defects in User Stories (within Dev Team control).
- Automated release test suite passes.
- Build is successful on all platforms and automated sanity test passes on all platforms.
- Legacy defects, or new defects from the waterfall side of the business follow the fire lane process.

Figure 22: The Definition of Done used by the BIOS teams evolved to what you see here after a few Sprints. This example Definition of Done along with additional context can be found in Table 9.3 of Forging Change.



*Figure 23: In *Scaling Lean and Agile Development*, Larman and Vodde talk about organizational strategy in terms of Jay Gilbrith's star model. Some of the greatest problems in the LeSS adoption relate to Rewards and People, and their intersection with Structure. There was a failure to flatten the formal organizational structure, coupled with insufficient attention to compensation structures which reward technical proficiency at the same level as managerial proficiency. It is useful to keep Gilbrith's model in mind as you read through the BIOS Alignment to LeSS Rules section. (Figure used with permission).*

Example Triage Guidelines

Version: 0.33

Date: July 4, 2017

Add to the **Sprint Backlog**, and start work immediately if:

- I see an email from partner company regarding future platform with subject [Blah-blah] or [Blah-RC]
- I see an email from build system team, build is broken
- I see an email indicating the automated long-duration system test failed
- I see an email from the partner company with Future Platform label notification
- Platform team believes delaying the work will affect hardware schedule
- The issue is known to be blocking manufacturing

Add to the **Sprint Backlog**, and start work after approval (PO will likely say yes) if:

- I get an email from partner company regarding a shipping platform with the subject [Blah-blah] or [Blah-RC] and I then convince myself this is a critical release that needs immediate attention and cannot wait until the next Sprint
- Severity 1 or Severity 2 bug
- Issue is raised by manufacturing, but not a blocking item
- Issue is raised by HW/[Sub-System X]/[Sub-System Y]/[Sub-System Z] teams, but not a blocking item

Add to the **Product Backlog**, we'll prioritize during Product Backlog refinement meetings if:

- I get an email from partner company with Shipping Platform label notification
 - I get an email with "+Bob" without clear indication of urgency
 - Someone says: "Hi, we need Sam or Sally's help with ..."
 - New Severity 3 or lower bug is reported.
 - Asked to work on old Severity 3 or lower bug
 - Platform issue reported, but is not bound to an immediate date
-

Figure 24: This triage guideline used by the BIOS teams establishes three basic categories: take immediate action, ask the Product Owner, and put it on the BIOS Component Backlog. The Product Owner has clearly communicated intent while empowering the Scrum Development Teams to take immediate action when appropriate. By making the guidelines explicit, the Product Owner also improves the odds of collaboratively refining the guidelines based on the collective wisdom of the teams.

Conclusion

Reflections on Deep Organizational Change

In retrospect the early loss of the supportive engineering SVP coupled with the lack of long-term job safety in a culture of frequent layoffs focused on short-term quick fixes of profit margins drastically reduced the odds of long-term success in transforming the organizational structure and culture. Without these challenges I believe success within the BIOS component could have been leveraged into a broader LeSS adoption.

After the success of the LeSS adoption within BIOS, I believe the original engineering SVP would have helped achieve the broader enrollment of senior management that was required to go further. Instead the senior management was busy scrambling to push out the upcoming MCS release while struggling to navigate the political landscape of an impending layoff. I believe more aggressive structural changes earlier on would have been very helpful, yet I don't think the senior management involved would have been ready to accept it without the time we took to slowly build up a record of success.

I am reminded of the parable of the sower in which a farmer casts seeds broadly over a variety of soil conditions. Although the seeds cast on shallow soil germinated and grew quickly, they soon withered under the sun for lack of deep roots. In contrast, seeds cast on deeper soil produced an abundant crop. Failure to properly prepare the soil for a LeSS adoption is likely to result in problems over the long-term, even though you will often encounter dramatic success early on. The soil for the BIOS LeSS adoption was sufficiently deep to produce some great early results, yet not quite deep enough to reach its abundant potential.

Summary of Benefits

The BIOS teams started as extended component teams while moving towards *expanded* component teams and feature teams. Benefits included:

- **Locally improved adaptability and value delivery within the extended component boundary**
 - Prior Observations:
 - Only one or two people understood a given BIOS sub-component. Code ownership and knowledge was effectively at the individual file level.
 - Knowledge of the codebase outside of the BIOS component was almost non-existent within the BIOS engineering group.
 - No one person was aware of the entirety of the work required to bring up a new Intel CPU generation on new blade hardware. Each BIOS firmware engineer and BIOS firmware tester only understood their small piece of the puzzle.
 - Prior Impacts:

- BIOS engineers could not choose the most valuable work at any point in time, because they were restricted to their individual area of expertise.
 - It was nearly impossible to accurately forecast completion.
- Subsequent Observations:
- Every team member was slowly becoming comfortable working on a broader number of BIOS sub-components than they previously were.
 - Many BIOS team members were becoming increasingly skilled in testing and modifying every aspect of the overall system along the execution path between the GUI and the BIOS component.
 - Every BIOS team member had a good overall understanding of the work required to bring up a new Intel CPU. Any member of any of the LeSS-oriented BIOS teams could explain any item on the BIOS component backlog and how it fit into the overall picture.
 - The BIOS component backlog consistently provided an up to date view into all of the known remaining work required to ship the upcoming BIOS release.
 - The BIOS teams accepted the natural variability of product development, and committed to *quality* (not *scope*) as explicitly expressed in the BIOS Definition of Done.
- Subsequent Impacts:
- The LeSS-oriented BIOS teams were able to consistently focus on the most valuable work at any point in time.
 - The BIOS teams were far more confident in their release forecasts.
 - The BIOS teams were able to ensure a releasable level of functionality was achieved before working on lower priority functionality.
 - The LeSS-oriented BIOS extended component teams ensured every Sprint Increment was successfully integrated and tested with the MCS system as a whole. Across a few thousand engineers, only the small Diagnostics pilot Scrum team could claim anything similar.

- Improved technical practices that created **improved quality**, along with improved awareness of what additional improvements could bring
 - Prior Observations:
 - There was no common agreement on what the quality standards were.
 - Changes in the BIOS code supplied by AMI were co-mingled with the MCS specific code in a difficult to maintain fashion. Every new generation Intel CPU effectively required starting from scratch.
 - Prior Impacts:
 - Adapting MCS to each new generation of Intel CPU took over a year and an army of BIOS engineers. Failure to complete this work before Intel officially released a new CPU would result in tens of millions in lost revenue.
 - Subsequent Observations:
 - The BIOS Definition of Done is a living quality standard embraced by each of the new BIOS teams. The Definition of Done is continually clarified and extended over time by the teams themselves.
 - The BIOS Definition of Done has always included a commitment to ensure any MCS BIOS changes leverage the AMI plugin layer whenever possible. This sensible requirement originated within the teams, not from the coach or management.
 - Subsequent Impacts:
 - Improved technical practices are expected to help dramatically reduce the time required to adapt to future generations of Intel CPUs.
 - Healthy retrospectives and a focus on quality continuously provide clarity on the next improvement.

- Early identification and resolution of **defects** related to the extended component
 - Prior Observations:
 - BIOS firmware engineers would make code changes and then throw the code over the wall to the testing group. It would generally be weeks or even months before the testers would identify problems and throw them back over the wall to the BIOS firmware engineers.
 - There was very little if any automated testing. Almost all testing was very manual.
 - It was common practice to commit code to source control which had compilation problems.
 - Prior Impacts:
 - It generally took months before defects were found and resolved.
 - BIOS engineers often wasted time chasing breaking changes introduced by another developer. Many times this was done by another developer in a distant time zone who was no longer at work when the problem was discovered.
 - Subsequent Observations:
 - Every BIOS team contained people with the skills and domain knowledge to both develop and test any changes.
 - Each BIOS team typically focused on making a small number of changes at any one time. They worked as real teams rather than as individuals as they actively focused on achieving the Definition of Done for the relevant Product Backlog Item.
 - BIOS team members actively cross-trained each other in both technical and domain knowledge. Testers became stronger developers, and developers became stronger testers.
 - Increasing levels of automated testing, and automated build systems helped further ensure problems were discovered as early as possible.
 - The automated builds and tests were especially helpful in identifying conflicting priorities between the legacy teams and the LeSS-oriented teams.
 - Subsequent Impacts:
 - Defects were found and resolved within minutes or hours rather than months. The usual months wasted in multiple months of bug squashing after a “code-freeze” were completely unnecessary.

- Conflicting quality incentives and goals of the overseas teams working in the legacy structure became extremely visible once the overseas teams started to work on the same codebase. With time and management continuity I believe this visibility would have made it politically viable to restructure and improve the overseas teams as well.
- Improved employee **collaboration, engagement, and learning** within the extended component teams
 - Prior Observations:
 - People worked as individuals rather than team members, each person focused on their own personal assignments.
 - The teams followed a typical engagement model in which the manager led and drove the interactions and work assignments.
 - Growth and learning was limited to an individual’s area of responsibility and historical expertise. Expanding one’s expertise and responsibilities into new areas required arranging and shifting assignments. This took time, and was not always practical.
 - Prior Impacts:
 - “A lot of effort with limited returns.” – Mitya
 - Subsequent Observations:
 - The teams managed the work themselves, collectively and enthusiastically focusing on delivering the PBIs their team selected.
 - Individuals and teams actively learned whatever was required to complete the PBI their team was focused on at the moment. The degree of knowledge transfer within and across teams was an order of magnitude greater than previously observed.
 - Subsequent Impacts:
 - In the terminology of *Leading Teams* by Richard Hackman, BIOS teams acted as *real teams* rather than loosely collaborating individuals. Each team had a clear team *task*, clear *boundaries*, clearly specified *authority* to manage their own work processes, and membership *stability* over time.

- Increased awareness of **organizational impediments** and the need to make even more organizational changes
 - Prior Observations:
 - Long development cycles, late discovery of defects, frequent hand-offs cross teams, work becoming bottlenecked on a single individual, and lots of other problems were accepted as normal. There was very little awareness that things could be significantly better.
 - Individual team members had little structural incentive to focus on adaptability or value delivery at a global level.
 - There was very little discussion or effort taken to remove long-standing organizational impediments.
 - Prior Impacts:
 - There were very few productive efforts at improving adaptability and value delivery at a global level.
 - Subsequent Observations:
 - Regular team and overall retrospectives within the BIOS teams frequently identified organizational impediments, which the teams and management would then actively attempt to resolve.
 - The efforts of the BIOS extended component teams to fully validate any BIOS changes increasingly drove them to challenge code ownership boundaries and expand their ability to make code changes across multiple components.
 - The pursuit of more rigorous quality standards manifested in the BIOS Definition of Done helped highlight the conflicting structural incentives of the legacy BIOS teams.
 - Subsequent Impacts:
 - People were slowly waking up to what was possible, and the need for greater organizational change.

Wrapping Up

I strongly believe the best chance of long-term success is achieved with a broad product boundary coupled with feature teams executing within the full scope of the product. If the art of the possible excludes this choice, an incremental approach starting with extended component teams and aggressively moving towards expanded component teams and feature teams can still be worth pursuing.