



CollabNet Success Story: Intel Corporation

AGILE PROJECT DEVELOPMENT AT INTEL: A SCRUM ODYSSEY

By Pat Elwer, Intel Corporation. Contributors included Tim Gallagher, Intel Corporation; Katie Playfair, Dan Rawsthorne.; and Michael James, CollabNet

ABSTRACT

In the microprocessor industry, the product development engineering (PDE) group exists to provide the test collateral to support cost-effective device screening and classification. Squeezed between the actual design teams and factory manufacturing teams, PDE is often put under tremendous pressure without ultimate control of team level deadline, scope, requirements, or deliverables.

To better coordinate the efforts of the sub-teams within PDE, seven teams comprising approximately 50 people volunteered to pilot a more integrated approach to product development. To organize this integration, the authors decided that Scrum was the best project management framework to employ along with agile engineering best practices. This paper describes the journey taken by the organization, its lessons learned, and the results of its investment in Scrum.

INTRODUCTION

In the software engineering world of agile and Scrum implementation, a plethora of best practices exist. Many of these involve small or medium-sized organizations using small teams, developing software in object-oriented languages. The Oregon and Pacific (OAP) PDE team is a large organization needing to implement Scrum and agile across multiple teams, sites, cultures, and environments. Our work product is a Test Program which runs on Automated Test Equipment (ATE). ATE has a proprietary operating system and interface languages which prevent us from using industry standard, off-the-shelf software validation solutions. In essence, we are working in a proprietary language environment with no off-the-shelf unit test framework and no offline testing capabilities. Additionally, we have a long history of requirements thrash, over-committing, missed schedules, insane work weeks, poor morale, and high turnover rates.

A long history in fabrication and manufacturing has resulted in a strong waterfall culture at Intel, embraced widely as the best path to success. Teams are organized in functional "silos" with regularly scheduled hand-offs of deliverables to other functional silo teams. The result is that some teams carry an unusual burden at late phases in the lifecycle and had very high turnover at the end of a project. Finally, each team comprises domain experts whose skills seldom overlap with those of their team members. It is therefore difficult to impossible to pair on tasks or share responsibility within a team.

Intel Corporation
(NASDAQ: INTC)
Santa Clara, CA
www.intel.com

Founded: 1968

Employees: 86,300

Annual revenue: \$38.3B

Products: microprocessors, flash memory, motherboard chipsets, network interface cards

Despite all of these challenges, we wanted to move ahead with a different way of managing and organizing projects to better unite test teams and smooth out the delivery of our work product. We chose to introduce Scrum to the organization at the outset of a project, when most of the work was development of pre-silicon infrastructure and readiness work. If we could get Scrum to work during this first phase, we felt the best practices learned in this relatively calm period of the project would find their way into the more stressful execution phase—when daily work is dependent on the health of actual silicon, dynamic external business conditions, and the requirements of the Fabrication, Design, and Manufacturing customers.

PHASE 1: PREPARING FOR SILICON

The initial transition group included six teams and numerous sub-teams. As a first step, we retained CollabNet as a Scrum education and coaching vendor. Approximately 20 group leads and technical leads attended a two-day Certified ScrumMaster training course as an intense introduction to Scrum principles and practices. Unfortunately, three senior managers missed the training and this resulted in subsequent impediments throughout the transition process. Executive sponsorship was critical to our success. Having our three most senior leaders absent from the initial training led to gaps in their knowledge of the changes we were trying to make.

After the training, participants attended a retrospective meeting and discussed, without CollabNet representatives present, their thoughts, reservations, and commitment level to a Scrum approach to project management. The team leaders agreed to commit to three months of implementing Scrum principles and practices “by the book” prior to questioning the effectiveness of the new process or attempting to tailor it to Intel needs. A Process Action Team (PAT) was formed to monitor the development of Scrum within the pilot teams and to provide support for process questions. Even though agreement was there, I could already sense a split in the organization into “pigs” and “chickens” in terms of supporting Scrum.

Group and team leads functioned as the Product Owners for all seven teams, while I worked as the ScrumMaster. I felt strongly that Scrum was an important framework to implement within these teams and I was willing to take the risk to champion it. Although the result of taking this risk was positive, I nearly didn’t survive a full quarter of ScrumMastering seven teams!

Working with CollabNet consultants Michael James and Dan Rawsthorne, we determined that having volunteer ScrumMasters would be important to each team’s success and to their own sanity. First, we worked with Intel management to make sure that the role of ScrumMaster was valued in the performance appraisal system as “real engineering work,” rather than administrative overhead. Secondly, those who stepped up to take ScrumMaster roles did so on teams in which they did not have a technical stake. This helped prevent any conflicts of interest between their own technical projects and their facilitation responsibilities. Budget did not exist for ScrumMasters to give up their own engineering work in favor of ScrumMaster work. However, support was lent for a lighter change in product role for those who stepped up to shepherd the process in ScrumMaster positions.

At the end of three months, there were three additional ScrumMasters to manage seven teams. Additionally, an eighth team had volunteered to start using Scrum.

After approximately five months, scaling work across the Scrum teams became one of the largest challenges. Before adding the additional five

teams that were formed throughout the remainder of the year, the organization needed more knowledge on how to manage the dependencies between multiple teams and facilitate better inter-team communication. Danube was again retained to develop a customized Scrum scaling course for some of the original participants of the Certified ScrumMaster course along with the senior managers who missed the first class. This day-long training reviewed major principles of release planning, sprint planning, and, in particular, scaling across multiple teams. We again took the “learn, try, inspect, and adapt” approach to this scaling.

After learning a few “best practices” for scaling, we took the issue back to the teams to try one of the scaling models from class and then tailored the approach to the teams’ real world environments. After adding some roles to handle technical dependencies and more layers of organization, the group was able to scale to 12 Scrum teams, each containing approximately five to nine developers, within a year.

Two of the primary aspects of successful organizational transitions that we discussed with the CollabNet consultants were volunteerism and self-organization. Although teams who committed to a three-month trial period of “by-the-book Scrum” were asked to adhere to the core principles and practices, adoption was clearly more important than adherence. A “please just try it” attitude from management resulted in better buy-in from teams. After the three-month trial, teams were given the freedom to organize themselves and inspect and adapt their approach every sprint. Although the teams needed to work together, they were given as much freedom as possible to determine what would work for them. Deviations were discussed, but not judged in the PAT meeting with all POs and ScrumMasters each week. Our goal at this stage was unity, not uniformity.

Visibility was also crucial to this process. An internal wiki allowed teams to document what worked for them, what didn’t work well, and suggestions for best practices for Scrum adoption.

Implementing Scrum “by the book” was an integral part of launching Scrum across the teams. However, at an organization the size of OAP, it is necessary to conform to certain organizational structures or requirements. After the three-month pilot period, some modifications were made to fit Scrum into our culture and environment. First, the team had to define which roles were most useful to their goals. We developed the following role descriptions:

- **Business Owners:** Senior managers or principal engineers charged with oversight of multiple teams or overarching technical issues for all teams. BOs set the roadmap milestones (Release Plan) and defined the ‘desired’ features at each milestone. Scrum teams still owned sizing and committing to meet the feature milestones based on their velocity.
- **Product Owners:** Typically functional group managers.
- **Technical Owners:** Technical leads from each of the functional areas who could collaborate on integration, dependency, and architectural issues to ensure congruence between teams with dependent outputs. TOs held ad hoc meetings to break down epics into sprint-able stories.
- **ScrumMasters:** A cross-team engineer with no specific stake in the project team he or she was ScrumMastering. This helped to curb the urge for the ScrumMaster to meddle in the technical solution.
- **Teams:** Team charged with one particular output of the test suite, rarely with cross-functional team members. Almost always a functional silo team.

- **Transient:** Group members with highly specialized skill sets needed by multiple teams for only a sprint or two at a time. They came and went at sprint boundaries.
- **Conduit:** Team member who represents more than one person including contractor supervisors or local members of a remote team. Conduits can sign up for many more story points of work than a normal team member.
- **Story Owners:** A technical expert with particular knowledge of how to complete a story who can develop tasks and request the participation of certain team members in completing those tasks. The one person you can go to and ask, “What’s the status of this story?” and get the right answer from, every time.

Finally, during this phase of product development, the overall group of Scrum teams was essentially its own customer. We were only building infrastructure to support silicon debug and manufacturing. There was no outside force requesting certain features during the first year or so of the project. This made business value a difficult metric for prioritization. Therefore, the POs and BOs tried to prioritize features with a combination of estimated business value and general priority, mostly as a dependency management strategy.

By the end of the first year, Scrum had taken root within the organization and become the default framework for planning our work and managing our requirements. The PAT had a wealth of data on which “tailorings” were and were not working. Silicon loomed on the horizon. Would the process hold up under real business pressure or would it get thrown out the window in favor of doing it the “old” way?

PHASE 2: SURVIVING SILICON

First silicon is a tough time to be a product development engineer. If you mapped it on the Stacey Diagram, it would be the most upper-right pixel in the chaos space! When silicon arrives, all requirements are ambiguous and it takes a few weeks to collect the necessary data on the silicon devices to determine the path the project will take.

I decided to step back and “inspect and adapt” my organization’s approach to Scrum based on what I saw happen at first silicon. What I saw really surprised me. I had one Scrum team that reverted to its old habits. A few other Scrums decided that they were done at first silicon and disbanded gracefully. The rest clung to Scrum like a drowning man to a life preserver.

Our two-week sprints were impossible to maintain in this environment and most Scrum teams went to one-day sprints instead. They would meet for one hour every day to plan the next 24 hours and review and reflect on the previous. Scrum’s four meetings collapsed under the gravity of first silicon into a single meeting. However, when I attended these meetings, I saw that Scrum’s core behaviors — such as business value based prioritization, team sizing, not working outside the backlog, peer updates and swarming, implementing process improvements and reviewing the work product — were all happening, just on a much smaller, faster scale as knowledge of the device was growing.

In the daily debug meeting, where all of the organization’s leaders and managers were in attendance, I would hear any ad hoc request being made followed by a PO saying, “Is there a story in the backlog for that?” I also saw many examples in which developers would grab a stakeholder or PO and drag them to the test equipment to witness that the content being added to their program met their acceptance criteria.

This period of intense debug and development went on for a few weeks. At the end of that time, the surviving Scrums emerged intact and expanded their sprints back to two weeks, like an accordion. The change was announced in the Friday Debug meeting, occurred over the weekend, and the two-week sprints remain in effect today.

PHASE 3: PREPARING FOR MANUFACTURING

As the silicon became healthier and we prepared for the manufacturing ramp, I noticed that the functional silo Scrums were being strained by handoffs in the post-silicon environment. A handoff occurs whenever responsibility, knowledge, action, and feedback are separated (Ward):

- One person decides what to do (responsibility);
- Another person defines how it will be done (knowledge);
- A third person implements it (action); and
- A fourth person validates the work (feedback).

Additionally, Conway's Law states that organizations which design systems are constrained to produce designs which are copies of the communication structures of these organizations (Conway). I knew that cross-functional teams were part of "by-the-book" Scrum and I felt that they would solve this particular impediment, but had not found a workable solution for forming them within the organization.

During this time, I noticed a number of Task Forces being formed to deal with silicon content issues. At Intel, a Task Force is a cross-functional team that is formed in response to a crisis. If you are tapped to join a task force, it's because you are an expert. You are immediately responsible for the success of the Task Force, you drop whatever else you are doing, and you can't say no. I couldn't see how to get the cross-functional benefit of the Task Force without changing the organization's structure.

I had coincidentally scheduled some Lean Product Development Training with Mary and Tom Poppendieck for the team in this timeframe and the Lean training revealed an important clue in how to use Scrum effectively in this phase:

- **Keep Functional Teams:** These are useful organizational structures as knowledge and deep technical expertise live here. They also give Scrum team members a place to "come home to" between projects.
- **Create Cross-functional "Feature" Scrums:** Functional teams loan responsible experts to cross-functional feature Scrum teams. Cross-functional Scrum team members are 100 percent dedicated and are not influenced by their functional managers during the sprint.

When I heard this, it really resonated with me. A cross-functional *Scrum is a Task Force without the crisis!*

We ran a quick pilot on one content type and the team members loved it. The handoffs had been greatly reduced. Team members were able to swarm on problems. Communication and knowledge flowed smoothly. And if a particular team member's function wasn't needed in a sprint, they paired up with another team member to cross-train and help where they could.

Once again, timing was everything. This cross-functional pilot data rolled in just in time for our annual offsite leadership meeting. This was a huge

opportunity to influence the organization's leadership and make a course correction that would allow Scrum to function even better than before. I presented my observations and the early data from the single cross-functional pilot and the organization was sold. In fact, all of the naysayers, as well as the undecided portion of the organization, signed on to the cross-functional Scrum concept. This added five new Scrums to our process, bringing our total impact up to 18 Scrums over two years!

RETROSPECTIVE:

I've used a simple (+) and (−) scheme to indicate what went well and what didn't.

Strong Definition of Done (+)

Since we don't have a 'real' programming language, we don't have a unit test framework or an offline regression. In microprocessor product development, unit testing means testing silicon units! This led us to focus on writing good stories and, most importantly, writing good acceptance criteria. Acceptance criteria (AC) provides a strong definition of done by detailing the requirements for customer satisfaction.

We also implemented a lightweight verification process that we called the "Pair Review." To complete a story, the developer and PO or stakeholder must sit together and agree that the AC have been met. We collected simple metrics on this activity in the form of Adds, Saves, and Escapes.

Adds are additional AC that the SH/PO add to the story during the Pair Review process that are accepted by the developer for the current sprint and are an indication of ambiguous story AC. Saves are bugs created in this sprint and caught in this sprint. Escapes are bugs created in a previous sprint and found in the current sprint. Saves indicate the verification process is working, while Escapes indicate that it needs to improve.

Additionally, we had to define a robust validation process. Validation couldn't be generalized across the Scrums like verification, so each Scrum documented its validation rules, essentially defining what it means to be done with validation for its work product. Validation must ensure the story will function properly in the released work product and usually involves running devices on the test equipment.

A story is only done if all tasks are done and it has been verified and validated.

No Partial Credit (+)

When determining velocity for the next sprint, no credit is given for stories that are not "done" based on our definition above. This may seem draconian, but was necessary to force the team to pay attention to the verification and validation requirements and make sure their estimates include these steps. It also forces the team to pay attention to its commitments. If you committed to 100 percent done and you delivered 90 percent, you failed.

Nine-day Sprints (+)

We sprint for nine days and hold Review, Retrospective, and Planning Meetings every other Friday. This way the team is always outside of a sprint every other weekend. This greatly helped improve the quality of life and morale of the Scrum teams. Conversely, every other weekend was in the middle of the Sprint and team members could decide among themselves if they needed to work the weekend to meet their goals. This happened rarely

after the first six months and we have achieved a repeatable cadence and a sustainable pace.

Cadence (+)

The nine-day sprint cadence allowed POs, BOs, and teams to change direction as necessary, at frequent intervals. This cadence actually helped reduce the requirements thrash that we had seen on previous projects and high-level managers began to see that the team was able to produce real work product every other Friday. Data we collected showed that 10 to 20 percent velocity was lost when a sprint was significantly interrupted. We called this the “sprint interrupt tax.” Ten percent of velocity was lost if the interruption came in the first week of the sprint and 20 percent if it came in the second week. Managers were made aware of this statistic. They began to respect the cadence of the planning cycles. We also added a rule that any change to an active sprint forces a renegotiation of scope. Again, management responded and when interruptions were needed, they usually came prepared to swap out items from the sprint backlog.

PO on the Team (–)

As a means of facilitating better communication between Product Owners and the team, we allowed Product Owners to serve as participating members of each team. In some cases, it worked quite well, but in others, POs micromanaged the teams, dictating day-to-day tasks, and impeding honest communication between team members. This has led to teams holding secret meetings to discuss real organizational impediments out of the view of their PO/functional manager. Although these situations appear to be resolving over time, they have crippled the team’s ability to self-organize. When we built the cross-functional Scrums, we disallowed this practice.

Central Scrum Tooling (+)

Scrum requires bookkeeping to generate useful metrics, like the burn down chart, every day. This is especially true when running multiple Scrums or Scrum-of-Scrums in your organization. Having a central, open-access tool contributed greatly to the success of the transition. When we started our journey, I couldn’t find tooling to support the Scrum-of-Scrums, so we created our own. We started with XPlanner and customized it significantly with Java and SOAP into something we called “XPlanner2.” Based on those learnings, we created a custom Windows application. This central tool has been a key enabler for managing multiple teams. While I believe that you must have tooling to enable and facilitate large-scale Scrum, the available off-the-shelf offerings have matured to the point which I probably wouldn’t take this homegrown approach again.

Huge Backlogs (–)

Managing an “all access backlog” is also a challenge. If anyone, at any time, can add anything to a team’s backlog, it can feel like the team is being bombarded with requests. Some POs wanted to lock down their backlogs which don’t allow for input from other team members or stakeholders. Our tool for Scrum puts “new” stories in a different pile than “accepted” Stories. The PO can then review each new story, discuss with stakeholders, and decompose the story appropriately. We also implemented a “freezer” for stories that we knew we wouldn’t get to for a few sprints. Stakeholders could see that their requests were in the freezer and the main backlog was only three to five sprints deep.

Story Points (+)

Since most teams didn't have a common frame of reference, most Scrums used ideal days. Relative size is better, but harder to get in most of my Scrums. We had to watch our use of the word "days" when talking with upper-level managers unfamiliar with Scrum. We quickly moved to talking about "points" when reviewing data with outsiders.

Tasks Take Less than a Day (+)

Throwing hour estimations out the window was liberating for the teams and managers. Stories are assigned a degree of "difficulty" in story points and tasks are simply items that are binary—either done or not done. A task is always less than a day, so if a person is working on a task for more than a day, we know they're probably impeded.

Burndown Observations in the Daily Scrum (+)

Metrics and visual representations of status were also vitally important to success to date. A visible sprint burndown chart proved effective in warning teams if they're falling behind and has prompted conversations with POs about status, prior to the end of a sprint. The ScrumMaster brought in the burndown chart every day. As a result, no one walks into a review meeting shocked that something did or did not get accomplished.

Incremental Review (+)

We didn't like waiting until the Review meeting to seek approval from the PO. Our Pair-Review verification process encouraged the PO and developer to sit together as soon as a story was deemed ready for verification. This eliminated most of the surprises in the Review meeting where the final work product was reviewed. It also made for a much shorter meeting.

Velocity (+)

Visibility of backlog, progress reports, and overall metrics help adjust manager expectations frequently so they can make business decisions based on the actual accomplishments of the teams. Velocity metrics force POs to schedule work to capacity. After all, you can't get 80 story points out of a 50-point team.

Executive Sponsorship (+)

Support has been a crucial win for both teams and managers. Without high-level support from the organization's manager, the transition wouldn't have been successful. Upper management provided structural support, incentives for those who took a leadership role on teams, and gave them career credit for their contributions. Leadership also provided disincentives for those who elected to subvert the process. Human impediments were "repurposed" to ensure success, but we didn't lose any people in the transition.

Changing Behaviors (+)

Finally, behavior is not learned unless it's practiced. My team has learned that the practice of Scrum begets better Scrum behavior and results. By consistently negotiating scope, practicing prioritization, authoring clear requirements, adhering to time boxes, keeping an eye on the metrics, and aiming for team self-organization, Scrum survives and thrives two years after taking our first steps.

RESULTS

Scrum has made an impact in four major ways: Cycle Time, Performance to Schedule, Morale, and Transparency.

Reduced Cycle Time

- Scrum was a major contributor to a 66 percent reduction in cycle time.

Performance to Schedule

- We have established and maintained capacity-based planning and a two-week cadence for more than a year.
- We have virtually eliminated schedule slips and missed commitments.
- Customers and upper management are changing their behaviors to protect the two-week cadence.

Improved Morale

- Improved communication and job satisfaction.
- What was lowest morale team is now best performing team.

Increased Transparency

- Led to adoption of formal, CMMI style, VER, and VAL standards.
- Scrum has uncovered bugs, impediments, weak tools, and poor engineering habits.

Scrum has been a major contributor to a consistent, repeatable, 66 percent cycle time reduction in the creation of our work product. While we also underwent some major tooling improvements, I believe that Scrum contributed on the order of 50 percent of those gains.

The nine-day sprint cadence provides robust schedule predictability. This predictability has actually led to less thrash in team requirements as management seeks to avoid paying the interrupt tax. We simply don't miss deadlines any more through aggressive management of priority and scope.

Job satisfaction comes from consistently hitting goals established with velocity-based planning. The team feels incredible pride in its ability to make and meet commitments. Morale is much higher and the sustainable pace is greatly valued in the organization.

Many, many traditional engineering practices and systems are being questioned as Scrum makes inadequacies more visible. This has led us to invest in additional infrastructure to allow us to adopt even more agile practices.

SUMMARY

Scrum has served us very well in Product Development Engineering. Word of our success is spreading across the company and I have been spreading the word on the benefits of Scrum.

We had many false starts along the way and had to learn a lot of hard lessons. However, we had strong commitment from our management and a tight cadre of Scrum believers that kept us coming back to make it better.

In the end, I think we have made great strides at changing our organization from a command-and-control, plan-based organization into an inspecting and adapting, self-organizing, empirical planning-based organization.

I was teaching an internal "Introduction to Scrum" class and a few members from my organization were in attendance. Halfway through the class, one of them came up during a break and said, "It's funny, but I didn't know there were rules. This is just the way I work." Changing behaviors is a long, hard journey, but worth the effort.

ACKNOWLEDGEMENTS

The author wishes to acknowledge the folks at CollabNet for their training, patient coaching, and problem solving when the need arose.

I would also like to thank Mary and Tom Poppendieck for giving us insight into the proper use of cross-functional teams.

BIBLIOGRAPHY

- *Agile Software Development with Scrum* by Ken Schwaber
- *Implementing Lean Software Development* by Mary and Tom Poppendieck
- *Lean Product and Process Development* by Allen Ward
- *Datamation Proceedings, 1968* by Melvin Conway

For more information, visit www.collab.net.

CONTACT US

Corporate Headquarters
8000 Marina Blvd,
Suite 600
Brisbane, CA 94005
United States
Phone: +1 (650) 228-2500
Toll Free: +1 (888) 778-9793

Portland, OR
Phone +1 (503) 248-0800
Toll-Free (US):1 (888) 532-6823

Chennai, India Office
Phone: +91 44 4220-3700

Shanghai, China Office
Phone: +86-21-61221082

Seoul, Korea Office
Phone: +82-2-722-8271

Tokyo, Japan Office
Phone: +81-3-5798-3101

London, UK Office
Phone: +44 (0) 207-397-8690

Munich, Germany Office
Phone: +49 (89) 24218-442