To Build an Agile Company, Do Not Begin with Agile Development

Michael Belding and Phyllis Thompson

mbelding@shiftwise.net and pthompson@shiftwise.net

Abstract

In small startup software companies, agility is survival. Companies that rapidly put limited resources on the most promising opportunities and customers are able to survive and grow. Companies in this stage do not typically have significant process and rely on people. In a small organization, engineering, sales, finance, and leaders communicate every day and freely share information. The focus on getting a product out to market where it can pay the bills, plus the ease of engaging decision makers, minimizes documented design work. If a startup is fortunate enough to grow, more people and the demands of customers will begin changing the landscape. People who used to communicate constantly now have more demands on their time. Newer employees have to fit into a culture where common cause is supplanted by specialized roles, and new programmers are expected to be instantly productive with limited documentation and limited training time.

At this stage small companies consider developing a process to address these competing interests while still growing the company. Agile methods such as Scrum, Lean, eXtreme Programming (XP), or Test Driven Design (TDD) arrive with a promise to provide just enough process to allow for growth while keeping the company agile.

This paper focuses on the experience of ShiftWise, a medium sized software company that grew from tiny startup to market leader, but also from an agile company to an agile development organization and then back to an agile company. It explores the cultural and business forces that both facilitated and hindered the simple goal of being an agile company.

Biography

Michael Belding is the Vice President of Engineering at ShiftWise in Portland, Oregon. Mike started working with Software as a Service back when providers were called "Application Service Providers" and has worked with Software as a Service in health care, transportation, finance, and corporate governance sectors. Mike has built highly responsive and innovative teams and coached them to deliver new products such as ShiftWise Connect, while continuing to develop and improve existing VMS offerings. Mike holds a BS from the University of Oregon.

Phyllis Thompson is the Agile Process Coach at ShiftWise, where she is has participated in the company's agile adoption since 2012. Phyllis has worked with agile teams for nearly 10 years in a variety of roles: Scrum Master, Agile Project Manager, Product Owner, and as the manager of a PMO (project management office) that rolled out an "Agile PLC" and Scrum/Agile training to more than 100 engineering team members at Serena Software, which used Scrum to develop an agile project management tool.

1 Introduction

In 2009, the technology leadership of ShiftWise was beginning to see signs that the company was transitioning from pure start-up mode to a more mature company. They began an evaluation of their capabilities and resources in preparation for addressing some issues that were preventing the company from making progress on new product offerings.

Initially ShiftWise had attracted customers who were early adopters of a new technology, which allowed them to automate their business processes, and they were willing to live with inconsistencies in delivery schedules and quality in order to get the new technology. However, after six years in business, ShiftWise was beginning to engage with new customers who had low tolerance for quality issues and much higher expectations for regular delivery of significant new features, and ShiftWise wanted to more confidently anticipate what the new markets needed and grow with those new markets.

The challenge was that the development team was collection of six individuals (five software developers and a QA engineer) who worked in isolation and did not think as a team. Each person had his or her own programming style, development philosophy, and own area of interest in the code. The company could attack five problems at once by assigning each problem to a developer to work on in parallel, but the efforts were fragmented. The result was an inconsistent and confusing user experience. There was a concern that perhaps half of each developer's day was taken up with responding to support issues, and a great deal of time was spent debating about what the issue was, whether it was an issue, and how to resolve it.

ShiftWise management felt that ShiftWise had a talented group of developers, but they were not cohesive. The leadership team in engineering had been with the company since its founding and were proud of the team's reputation for business awareness, flexibility, and dedication, but the group was managed one-on-one, not as a team. The company made frequent changes to the application, but over time, the majority of those changes were to fix issues resulting from other changes. It seemed that fewer new features were created. Plans and ideas to introduce major functionality were postponed in the face of the endless stream of change requests and defect reports. To grow, the company needed to change how it planned, built, and delivered software.

This paper is about the lessons ShiftWise learned as a result of growing from a company of twenty five to a company of over seventy and the journey from an agile startup, to a Scrum development organization, to a mature agile company. This journey represents the combined efforts of many, many individuals over the course of more than five years and the labor of an entire company to make it happen.

2 A Solid Foundation

The beginning of the journey proved to be an enormous challenge. The current development team had established the current collection of practices and it fit how they liked to work. They were open to growth, but there was no internal motivation to change the work dynamic. Likewise, the operations groups had adapted to work with the current development structure. Customer expectations, performance targets, and benchmarks were all keyed to how things currently operated. Thus, making significant changes would be disruptive to the business. The challenge was to drive change so ShiftWise could increase its ability to win the new opportunities the company was aspiring to, not to address impediments it currently had.

The first step on the journey was to find that aspirational path. Management started by identifying the impediments to growing the company.

The company leadership identified four significant impediments:

Lack of progress on customer feature requests

ShiftWise had a backlog of over 3,000 potentially actionable items, including defect reports, new feature requests, feature enhancements, and performance problems, but customers were becoming increasingly frustrated from what they perceived as a lack of progress on those items.

Complexity and inconsistency in the product

As the product evolved, changes were introduced to solve specific issues, or address very narrow requests. This meant that configuring the system required significant expertise to align a multitude of independent settings to create the desired experience. Many features leveraged settings used by other features, resulting in areas of the system that were unexpectedly linked in obscure ways. The net effect to the customer was a confusing application accessible to only a small portion of their employees who had been well trained on how the product worked.

Lack of visibility into release cycles

The business had no visibility into when features currently under development would be released. A popular example was a time when the Customer Support team found out about a feature release from a customer who was having trouble with it.

Inability to manage product development

The business wanted the ability to drive the development roadmap, and confidently set goals and market expectations based on that roadmap.

The one area the business agreed was going well, and was vital, was the flexibility and responsiveness that engineering provided to the Customer Support and Account Management teams. These business units had grown dependent upon being able to call a software developer directly if they needed to mollify a customer or to get an issue addressed quickly, and it was a capability they did not want to lose, even though this flexibility resulted in a disjointed product and generated difficult-to-diagnose defects.

It was clear that, in time, ShiftWise could develop a more process-driven approach to accomplish their objectives, but that any attempt to rapidly transform the organization or make significant changes was not likely to succeed because of the cultural and operational constraints.

The next step was to turn the list of impediments into goals that could be measured. The goals established by the company leadership were:

- The company must have the *flexibility* to pivot to new opportunities or respond to threats in a timely manner.
- The application had to have sufficient *quality* for customers to be confident in it and in the company.
- Decisions had to be made based on the *customer outcomes* to be achieved so that changes were met positively by customers and to ensure that work the company invested in was viewed as valuable.
- Customers and support teams needed *consistent* behavior both from the platform and the development organization, to have confidence that the company was providing the value they claimed.

3 First Moves

The next challenge was measurement. ShiftWise was, and in many respects continues to be, a results-driven company where successes and failures against goals are the principal metrics. Incremental or process metrics were not consistently kept. Before the company could change, management needed to know what the development team was actually working on at any given time, confirm if estimates for groups of tasks were accurate, and track where the work requests were coming from. Only then would the company understand the effect any change would have, and whether the changes were positive.

Management asked the development team to start measuring their work, but this request was not well received. This group of developers had established a set of individual practices that worked well for them and met the organizational goals as they understood them. While management could have imposed a requirement that the team begin tracking work, doing so would have created tension in the organization and an adversarial relationship instead of a productive one. There was even a risk that it might lead to attrition, and the company was not in a position to take that risk at the time. The company decided that transformational change would fail if it was forced or imposed. One of the managers likened the effort to taking a long journey with your family. If they wanted to go and were excited about the journey, they could tolerate a lot of hardships along the way. If they did not, you could anticipate a lot of “are we there yet?” comments. This idea evolved over time and became the first rule ShiftWise would establish to shape change management in the company. The rule became:

Bring people along on the journey

At ShiftWise, this has come to mean that whenever a significant change or endeavor takes place, the people driving the effort are responsible for ensuring that everyone who participates in, or is impacted by the effort, is fully committed to, and supportive of, the effort.

As noble as the idea was, the lack of transparency still remained and had to be addressed. It was clear from the reaction of the development team that a major change was unlikely to succeed so the management team settled on making smaller changes. They started working with two members of the team who were the most supportive of the effort. Those developers were asked to do their normal work routine, but every two weeks to list out what they thought they would do for the next two weeks. They also tracked their work in a ticket system, something nobody was doing with any consistency, so it was possible to see how they spent the time.

It was not a major change, but the journey had begun. After a month of tracking, the data showed that about 50% of the time, these two developers worked on what was called “Tier 3 Support” items, which were calls from the Customer Support department that were deemed serious enough to warrant immediate action. Issues ranged from running custom reports, to fixing bugs, to researching various behaviors in the application. Tracking all the work items validated and quantified the Company’s belief that a significant portion of development time (50%) was spent on operational support issues. Of the remaining time, about 25% was spent working on planned tasks and the other 25% was spent on ”refactoring” which generally meant rewriting the code of other developers so it met standards as they individually defined the term.

These two-week iterations demonstrated to the development team that engineering could add minimal process (“record what you plan to do and track what you did”) without the feared consequences, and more members of the team agreed to start tracking work. It also showed the business how little actual capacity the development team had for new feature work. This information gave the business insight into the schedule implications of various types of requests and the 50% of engineering capacity “tax” that the current production support burden placed on new development work. The development team was still not committing to complete specific deliverables, but the business could see work moving through engineering and it provided them confidence to more accurately message upcoming product changes to customers.

Based on this success, ShiftWise decided to continue the approach of making changes in small steps to avoid pushback, and adopted a second rule for the journey:

Move in increments

This rule goes beyond programming in increments, but also to making organizational changes in increments. It meant that each change had to be executed in a way that enabled the product planning, development, delivery, and support groups to internalize the change and the company leadership had to be able to measure whether the desired result had been achieved. It was a slow way to go, but it was important to keep all the groups positive about the journey while minimizing the journey's impact on them.

4 Major Changes

Over the next two months, engineering began to deliver on a predictable basis. The small process changes had given the business more visibility into development work in progress, and proved that engineering could deliver work more consistently. Equally important, because the changes affected only the 25% of engineering's capacity that was directed at feature work, the department's flexibility and responsiveness to customers was retained. Also, because engineering management had not significantly changed how the developers worked or were organized, the developers did not object to the effort. This incremental approach showed progress toward the goals without creating resistance.

These successes, and the care that was taken to maintain practices the company believed were critical to the business, brought broader support for the effort. The journey now included the executive team, operations teams, and the engineering team, and had demonstrated success with the goals. The company was ready to build on this success, but they still had what turned out to be an insurmountable obstacle: the development team members continued to work in silos.

There was a very important new feature on the roadmap that was so large that one individual could not deliver it in a reasonable amount of time; the whole team had to work together on it over multiple sprints. The management team focused on creating an environment where the developers could work together as a team and collaborate on a design to deliver the big new feature in a reasonable amount of time, but the developers continued to approach the work as individuals. The development team could not agree on the best way to divide the work, or which approach to take. In more than one case, a developer settled an issue by just writing a block of functionality as they saw fit so they could claim that part of the project before the team reached consensus. The designs changed often and the total effort ballooned into a nine-month project. Eventually the cost and the disruption the effort had created became greater than the value it offered and work on the feature was cancelled.

The leadership team regrouped to consider how to proceed. ShiftWise was committed to growing the company, believed that the journey was the right one, and were pleased with the initial results. They valued the development team, but the spectacular failure while attempting to build a large-scale feature presented them with a dilemma. ShiftWise could keep the people, or continue the journey toward growing the company; there was no obvious path to get both. To grow, the development organization had to either work as a team or the company needed a development group that could. The company believed that with continued effort, it was possible that the developers might be compelled to work together, but that they would not choose that path, and to attempt a top-down change would be a strategy without any predictable end point. The company determined the most viable path was to replace the development team.

This surprising step gave birth to the third rule:

Be able and willing to make big moves

No organization would consider something this drastic without weighing the cost, consequences, and disruption that would occur. This big move was not capricious or impulsive. ShiftWise had embarked on a journey to facilitate growth and evidence that a radical change was necessary was clear after six

months of data collection and also from the six years the executive team had spent in the current organization.

In keeping with the previous rules ("bring others along on the journey" and "move in increments"), the replacement of the development team had the support of the company's leadership. To help manage risk, they planned the transition to happen over twelve months,

The leadership team hired an experienced Agile Product Owner and immediately replaced three development team members. The remaining three development team members were offered incentives to stay on during the transition. The engineering manager hoped that the three remaining developers would integrate into the new organization and stay with the company. Over the twelve-month transition period continued issues with collaboration, transparency, and trust forced the company to abandon the effort to integrate the remaining original developers and instead complete the transition to an entirely new team.

The next task was to address the company's 3,000 item backlog and a separate bug tracking list. These represented every known new feature request, bug report, and idea captured over the last six years. The difficulty was that there was little consensus on the correct behavior of many features. Getting agreement on whether something was a defect or correct behavior took considerable effort. As for feature requests, it was not possible to determine which described capabilities ShiftWise wanted to introduce, which features had been superseded by other work, and which would still be valuable to customers.

Rather than attempt to reconcile these lists, the executive team elected to discard them. At that moment, the application had no baggage; it was doing exactly what it was supposed to do, and was working as it should. Going forward, all new feature requests, change requests, and defects would be written as stories in the form *[As a <type of user>, I want <some goal> so that <some reason>.]*.[1] An unintended but beneficial effect was that all stories were now easily consolidated into a single prioritized backlog that the new Product Owner could manage. ShiftWise used stories to describe the desired behavior of the working application rather than documenting what was wrong with it as a defect, and eliminated the bug tracking list and the time-consuming bug scrubs that went with it.

5 Transition to Scrum

Throughout 2010, the company focused on the transition and building up the new team. After discussing what processes the new team should adopt, the company decided Scrum best fit the journey. Scrum allowed the organization to work incrementally, encouraged collaboration, and had a wealth of available resources. The whole team went through an in-house Scrum training and spent the first 90 days working on small items to quickly build up experience. The Product Owner presented a body of work for the team to address. The team provided estimates and then worked in two-week increments, first doing development and then doing testing, until the work was completed and ready to go out at some future date. The work was time boxed, but it was acceptable for tasks to carry over between sprints, and for the application to be in a broken state at the end of an iteration. Less than six months after the transition began, the new team delivered its first significant feature, a document management module for the ShiftWise core application. The feature was delivered to the market on schedule and was so successful that it was positioned as a new product offering. It was the first significant new functionality in almost two years, and was the first significant validation of the ShiftWise agile journey.

6 Full Scrum

In 2011, ShiftWise embarked on a new green-field product, and a rapid expansion of the development team.

For this new project ShiftWise followed a strict interpretation of Scrum. The company put all new hires through Scrum training, practiced the prescribed Scrum activities, and implemented recommended Scrum and XP practices. Work on the new product started right away because the company leadership trusted

that by moving incrementally they could concurrently manage the complexities of architecting a new product, reconciling coding styles, and communicating business goals.

Although it was not immediately apparent, the decision to dive into pure Scrum had some significant consequences:

- The team adopted Scrum measures for velocity, acceptance criteria, test coverage, and productivity. Because the company did not have a history of other measures, the Scrum metrics were heavily relied upon to guide, validate, and provide course correction to the overall effort.
- ShiftWise historically based hiring decisions primarily on technical skills and believed that the culture would evolve naturally when smart people interacted within the Scrum framework.
- The Product Owner established a feature set that was considered “minimally viable” to present to early adopter customers and the teams established a narrow “definition of done” that gave them confidence that they were meeting objectives as measured by the Scrum metrics.
- ShiftWise rigorously followed the Scrum process and inferred that the business goals were met because the Scrum measures looked good, rather than establishing metrics that measured the organizational goals they set at the beginning of the journey. For example, sprints could be closed even if the application was not fully functional as long as all the stories passed the stated acceptance criteria.
- Lack of a working application prevented non-programmers from gaining experience with the product as it took shape. The Product Owner gave demos of the features, but the operations and support teams were not able to gain a deep understanding of the new product or contribute concrete feedback about what might be missing. They were disconnected from the journey.
- The Product Owner was the single accountable authority on prioritization of work and drove all development efforts from the prioritized backlog of stories. This was generally a positive improvement; however, as the target release date neared and too much work remained, pressure built to prioritize the customer-facing features at the expense of features regarded as “back end” or “non-customer facing.” The effect was an application with very complex features, but few tools to manage them or diagnose post-release issues.
- Everyone trusted that by following the Scrum process rigorously they would get the results they expected.

In the fall of 2011, with high confidence provided by the Scrum metrics, but without customer or business validation, ShiftWise launched the large green-field product to early adopter partners, and right away they identified serious issues. Engineering had validated and tested each component, and correctly assembled the application in its minimally viable state, but the modular development approach did not require all the elements to function as a whole until late in the process. The organization had optimized for the most efficient development process at the expense of ensuring that every feature was implemented completely and did not need additional work post-launch to ensure that the whole application was supportable and maintainable. At launch, the new product was overweight on customer features and some of those features were overbuilt and therefore excessively complex for the audience.

The focus on features at the expense of back-end tools meant that the application was thin on diagnostic tools and other utilities that could give the Customer Support team insight into how the application was working so they could help customers who had trouble with the complex features. It was very common for users to make mistakes setting up the new application and there was no easy way for Customer Support to tell what was wrong. Thus it was difficult to determine when a problem was due to a misconfiguration versus an application defect.

Because the whole company had not participated in the journey, the delivery and operations side of the company was not ready to support or mange the product when it came out. Customer frustration grew as more and more issues emerged and confidence in the product faltered. Eventually the Sales organization decided to pause the rollout of the new product and the company regrouped.

7 Agility

It took six months to settle the issues with the new product, and another year to evolve the features to find a place in the market. This experience of initial launch using Scrum exclusively led ShiftWise to add a fourth rule:

A working application is the measure of success

This rule shifted the measures from process to outcomes. In practice, engineering and stakeholders became accountable to the state of the application, not the mechanics of the methodology. Velocity, unit tests, and test coverage still mattered, but only as benchmarks for the team, not as a measurement of application readiness for launch. Having access to a working application empowered the stakeholders to decide if they could launch and support the product in its current form, or if not, to see clearly what work remained.

ShiftWise now needed to find a way to apply the goals of the organization and also use the rules that resulted from recent experiences. The company elected to stay with Scrum as the development team methodology, but made a number of changes to align to the goals and keep to the rules:

Shippable code

All sprints now had to end with a shippable application. If a feature could not ship in a working state because it was not yet complete, it had to ship disabled. This did not mean that every sprint resulted in a deployment to production, but if the company wanted to ship a feature at the conclusion of a sprint, engineering had to be able to do it.

The Transformation of QA to SDET

To enable a deployment to production at the end of each sprint, all testing had to be accomplished within the two week sprint. The existing QA model did not allow for this so ShiftWise decided to transform QA. Instead of a dedicated QA department that tested after development finished, ShiftWise made the development teams accountable for testing and embedded Software Development Engineers in Test[2] (SDETs) with each team. The SDETs are automation specialists and team quality coaches. The development teams as a whole are accountable for the quality of the work, not a separate QA department.

Milestones to define a shippable application

To address the gap between meeting acceptance criteria at the story level and having a minimum marketable (viable) application, ShiftWise added the concept of milestones to the process and created a cross-functional group called the Design Team to define and monitor the milestones.

The Product Owner has a significant voice on the Design Team, but it is balanced by empowered advocates for the application, including representatives from operations, IT, and the application architects. The Design Team members collectively determine what will be required to ship and support the product, taking into account concerns about scalability, security, supportability, and maintainability.

The use of milestones grew out of the company's past experiences with "minimally viable products." Milestones were created to force continuous review of the shippability of any large green-field effort. Today, any new effort that requires more than four sprints to be commercially viable (i.e., has enough functionality to sell), must be broken down into milestones. Milestones define what an application needs to achieve in order to be shippable with the feature set requested by Product Management. There can be one or many milestones before a minimally-viable state is reached, and the milestones inform decisions about future functionality by exercising the deployable, working application.

> Each milestone by definition is a "definition of done."[3] A milestone cannot close until the team can demonstrate, using the working application, that it has met all the milestone goals. In addition, the team must demonstrate that goals from all previous milestones are still being met. For example, a milestone could include back-end work for scalability that is not apparent in the story-level description of the functionality. Use of milestones prevents ShiftWise from putting too much emphasis on the front-end features at the expense of having a viable, supportable application at launch.
>
> At the end of each milestone, Engineering must be ready to put the application into production if the Design Team determines that the application is shippable at that point, and the business wants to release it because the minimally-viable feature set has been achieved.

In keeping with the rules, these changes were slowly implemented throughout 2012 and 2013 and into 2014. During that time the engineering organization grew from two to five agile teams and in early 2012, ShiftWise hired an Agile Process Coach to help the teams with the transition to end-of-sprint releases and to ensure continued attention to incremental process change while the organization grew.

The first new obstacle was strong resistance from the development teams to shipping at the end of each sprint. They did not believe that they could test the application well enough to ship on such short intervals. Past practice had been to package as much functionality as possible into a release because the release process itself was expensive and time-consuming, requiring a lot of complex and error-prone manual processes. Early on, the mandate to ship at the end of the sprint forced the teams to invest a significant portion of the sprint in testing and release work. Over time, the teams adjusted and rather than commit to large stories, the teams committed to much smaller stories with more effort focused on the mechanics of learning to continually release rather than delivering functionality. During the first "release" sprint, the team took four days of the two-week sprint to prepare for deployment to production. Within six sprints they had whittled the prep time down to less than half a day, and the quality of the work improved as they became good at releasing, which created another rule, the fifth:

If it's hard, do it more

In the first quarter of 2012, ShiftWise released the application once. After implementing the requirement to ship at the end of each two-week sprint, they released 33 more times in 2012 and 84 times in 2013. For the first half of 2014, the release count was 31.

Before the switch to a two-week release cadence, all ShiftWise releases required several weeks of post-launch stabilization, which included prioritized bug fixes and feature changes, followed by an increased support volume as the feature was adopted. Since the switch, the post-release stabilization period has vanished and ShiftWise no longer sees any increase in support queues post-release. In 2012, ShiftWise required two developers to work full time on Tier 3 production support issues. That workload has decreased to the point where ShiftWise now has one developer who works on production support issues part time.

Some production releases still have post-release issues, usually resulting from the complexity of the manual deployment process. Through 2013, about 17% of all releases had errors that were primarily caused by problems with a code merge or a configuration change between environments. In 2014 ShiftWise has made significant progress in automating this work, which has reduced issues related to configuration errors and merging of code.

A faster pace of releases, but smaller feature list, allowed the business to re-engage and return to the journey. Another benefit of frequent releases was that business always had something to talk about with the customers. Even if the value was small, the customers perceived that ShiftWise was listening to them. The original goal, to remain responsive, was served. The customers perceived value because they were seeing progress, and now the shipped features were more consistent and of higher quality.

As the company has evolved, many of the ideas they held about what makes a great team have evolved as well. Moving from "no process" to "Scrum" to "agile" has been rewarding, but it has been taxing on some team members. ShiftWise's brand of agile demands that developers take greater responsibility for

the quality of the application. It requires features to be broken down into small chunks that require coding practices that might be less efficient than if larger stories were planned and delivered, but the goal is not to deliver more, but to deliver software with higher quality and on a consistent cadence.

The engineering teams are coached to understand and express ideas in business terms rather than technical terms and to invest in activities that increase their confidence that the application is shippable, not just that they met the acceptance criteria of a story.

Over the years ShiftWise has discovered that while they have benefited from the contributions of some great technical minds, they have seen far greater returns from individuals who thrive in the ShiftWise Agile environment, even when new hires have to be taught the development technologies and architectural patterns. In multiple instances, ShiftWise has paid a price in attrition, poor quality, low team morale, or cross-department friction when they made hiring decisions based more on technical skills than cultural compatibility. This realization formed the sixth, and last rule:

Culture Matters

Simply put, this rule means that when ShiftWise recruits or promotes employees, the attitudes and cultural values of the people come first, and ShiftWise accepts that this may require investment to close any skill gaps that may result.

These experiences, and the learning from them, have enabled company buy-in on a few additional significant moves:

Product Owners and Engineering share responsibility for the application

ShiftWise has product managers/product owners, but they function more like product marketing managers. The application is owned by the development teams, who roll up under a team lead, who is ultimately accountable for ensuring that the application as a whole can meet company goals. This allows the product managers to cover a larger portfolio and to spend more time in the market. The change has eliminated the more familiar Scrum Product Owner role that serves as a 'throat to choke'[4,] and has created a culture of shared ownership by stakeholders who are engaged and committed to outcomes.

Focus on application health rather than velocity

In 2011, ShiftWise continually measured team and project velocity and would adjust features to align a project burn down with target release dates. By moving to a two-week release cycle with features being delivered continuously, the company is able to see tangible progress and make decisions based on the state of the product. The regular pace of features being delivered has replaced a focus on burn-down charts with a focus on any gaps remaining in the product that prevent it from being launched. Large efforts contain milestones to force the company to continually review the working application and ensure that there is a balance between customer features, supporting tools, and production delivery infrastructure at the end of each milestone.

Prior to 2012, the engineering teams would typically spend 5% to 10% of their time in a sprint on quality-related work. As of 2013 and going forward, the teams are able to incorporate demonstrations, unit tests, UI tests, integration tests, and test automation into the sprint. They are also able to clear up technical debt[5] before it accumulates. The time spent on feature work has remained fairly constant, but ShiftWise has successfully transitioned from doing quality and stabilization tasks after a release to performing this work prior to a release.

Whole team ownership of quality increases quality activities

With the move to SDETs and whole-team, cross-functional ownership of quality, the effort to execute quality-related tasks is distributed throughout the team. This enables the team to complete the whole

scope of work within the sprint (development plus test), and also requires the team to meet quality commitments even if the SDET is sick or on vacation.

Investment in architecture

About 10% of senior developer time is spent on architecture and proof-of-concept work. In the past, this number was zero.

Increased investment in new tools and technologies

The quality standards and sprint cadence have allowed individuals and teams to introduce an array of new technologies and tools. Where ShiftWise was once 100% on Microsoft-built tools, today more than 70% of the code base is comprised of an array of languages and tools that are curated, taught, and nurtured by the teams, giving ShiftWise greater flexibility even as application complexity grows.

8 Conclusion

For the last five years, ShiftWise has been on a journey to grow as a company while retaining the agility they enjoyed as a startup. At the beginning of the journey, ShiftWise executive leadership laid out a set of goals that provided guidance for the most important capabilities the company needed to have at the end of the journey. The company established and maintained a set of rules that emerged as a result of the successes (and failures) along the way.

To achieve those goals, ShiftWise switched from a largely organic, minimal-process development practice, to a formal Scrum methodology. From those experiences, the company learned to leverage the formal Scrum development process as a foundation, but blend it with outcome-driven measurements that put the emphasis on what was completed, rather than what was in progress.

This model has allowed ShiftWise to create a product development cadence focused on frequent small releases with clear customer value, and has enabled engineering to double the time investment in quality management to 40% per team, while more than tripling the size of the engineering department from six to 24 in the past five years.

These organizational changes have reduced the time ShiftWise spends on post-release production support from 50% to 1.3% of overall development team capacity, and has allowed incremental enhancements and evolution of the product.

The goal of this paper was to share the story of scaling an agile company by beginning with clearly defined company goals, developing an effective project management process to organize work, building an organization that embraces those values, and providing feedback mechanisms that measure how well the organizational goals are met.

Each company's journey, goals, and rules will be unique, but the important thing is to define the goals and establish meaningful measurements to help stay true to them.

9 References

1 Cohn, Mike. 2008. “Advantages of the “As a user, I want” user story template” http://www.mountaingoatsoftware.com/blog/advantages-of-the-as-a-user-i-want-user-story-template (accessed August 3, 2014).

2 Definition of the role of Software Development Engineers in Test (SDET), from the Microsoft description of the SDET role on their Careers page http://careers.microsoft.com/careers/en/gbl/professions.aspx (accessed August 13, 2014).

3 Definition of Done, from “The Scrum Guide” by Schwaber, Ken and Sutherland, Jeff. 2013. https://www.scrum.org/Portals/0/Documents/Scrum%20Guides/2013/Scrum-Guide.pdf#zoom=100 (accessed August 15, 2014).

4 Cohn, Mike. 2009. “The Fallacy of ‘One Throat to Choke” http://www.mountaingoatsoftware.com/blog/the-fallacy-of-one-throat-to-choke (accessed August 3, 2014).

5 Ramakrishnan, Srinath. 2013. “Managing Technical Debt” https://www.scrumalliance.org/community/articles/2013/july/managing-technical-debt (accessed August 15, 2014).