Knowledge Management and Culture for Agile SaaS

Stephen Oppelt

Walden University

Abstract

This paper will detail the implementation of knowledge-based management in a Support department for a software-as-a-service that is developed by the Scrum methodology. The paper will assess how knowledge can be cultivated and employed as a strategic asset for the company and discuss how inter-departmental communication can be improved by instituting a cross-training program in order to take advantage of and distribute the knowledge possessed by individuals in the company.

Knowledge Management and Culture for Agile SaaS

SignUpForce is a software company that provides a suite of SaaS tools that empower meeting and event planners to quickly develop registration websites, host those sites, provide email services, deploy on-site mobile applications, and maintain registration databases. The company does not currently have an effective, comprehensive strategy for documenting its software or providing self-help support options to its customers.

There are a number of half-measures in place that attempt to address the challenges of knowledge management and support documentation, but collectively these do not meet the company's current support needs and are not scalable as the company continues to grow. The QA and Development departments do not maintain comprehensive technical documentation for internal or external use, citing the agile Scrum development environment, the fast pace of development iteration, and frequent feature updates as a reason for its impracticality. As a result, there is no product manual that provides reliable information about system behavior or best practices, and all departments rely on a sort of "village knowledge" to answer these kinds of questions.

Since there are few viable self-help options for customers, each issue that a customer experiences is more likely to trigger an email or phone call into Support and thus require the personal attention and time of a Support representative. As the number of customers increases, the number of calls for assistance will increase as well. As the number of calls increases, additional Support personnel will need to be added in order to accommodate them. As the number of personnel in the department increases, the cost of providing technical support to customers increases. It is entirely possible for the company to continue on its current course without changing the processes that underlie the delivery of assistance to customers, however it

will be necessary to ramp up the number of personnel answering the phones in Support as more customers are acquired. The company as a whole will be more profitable if the cost of providing support increases less proportionally with an increase in the size of the customer base.

The Company's Current Knowledge Management Assets

Apart from live telephone and email support, SignUpForce's current knowledge offerings include live training webinars, training videos, help documents, and a user community. The training webinars, training videos, and help documents are useful for the purpose of providing an introduction to the software and how-to guides for its major components, but do not provide quick, searchable answers to direct support questions. Release notes from each major development iteration are posted by the Marketing department on part of the company's main corporate page, targeted as advertisements for new features rather than as artifacts useful for providing support. The user community can be queried by customers, but the results may also include forum posts that consist of suggestions or even complaints mixed in with official solutions; its format also makes it awkward for Support to proactively use it to develop a storehouse of product information. In all cases, the effectiveness of these knowledge offerings cannot be tracked or measured in order to determine the degree to which they actually allow SignUpForce's customer base to find their own solutions and ultimately deflect calls into Support.

As the organization currently stands, the live telephone and email support are provided by the Support department. The live training webinars, training videos, and help documents are created and updated by the Training department. Training also oversees implementations for some of the product's more difficult configurations as well as providing custom professional services for an additional charge to the client. Individual questions posed in the user community

are fielded by the Support department, but the community itself is maintained by the Training department. The Support department currently reports to the Director of Support and the Training department currently reports to the Vice President of Marketing. The Support team currently consists of thirteen members: nine representatives, two leads, one manager, and the director; the Training team currently consists of three members.

Knowledge-Based Management's Impact on the Organization

The practice of gathering, storing, and maintaining an organization's knowledge must ultimately be performed in such a manner as to create a competitive advantage for and provide value to the company. In order for this value to be achieved, the knowledge that is being gathered must be used effectively; it is insufficient for knowledge to be simply collected and displayed, in order for its value to be measured, metrics must also exist in order to determine the impact that it has on the business (Coulson-Thomas, 2013).

Knowledge-based support can provide a number of distinct advantages to an organization. Knowledge-based management can reduce risk to the organization by providing a stable and compliant platform by which employees can deliver consistent information to one another and to customers, in turn building trust and confidence in the product and the team supporting it. With Support, Training, and Sales all drawing from the same wellspring of knowledge, the potential for Sales to incorrectly set customer expectations will be reduced. This is not to imply that a Sales team would knowingly mislead a customer, only to ensure that Sales is delivering a message consistent with information provided by Support and Training. Furthermore, Support is responsible for providing assistance on multiple products and there is a wide range in the level of proficiency that different support representatives have with some of these products; being able to draw information from a centralized knowledgebase will allow all

support representatives to quickly and accurately answer questions in areas where they may not necessarily be subject matter experts and will accordingly reduce call times since customers will be able to immediately reach someone who can provide assistance on the first contact. Knowledge-based management can improve the efficiency of training by allowing knowledge resources to be available to all personnel regardless of time or location; as the company grows into a potentially international organization or expands to additional offices, this knowledge infrastructure will facilitate the training of both employees and new customers on demand, around the clock, and in disparate locations. This wider distribution of knowledge will also facilitate the limited resources in training to be able to onboard larger numbers of learners (Coulson-Thomas, 2013).

Implementing Knowledge-Based Support

Implementing knowledge-based support in an organization is not as simple as writing a new process document; a fundamental change in the way that knowledge is modeled within the organization will require not only the implementation of processes, but also the deployment of new tools and structures and the re-training of stakeholders who are directly involved in the existing processes. Generally speaking, the stakeholders identified as part of the knowledge-based support process are the adoption team that is responsible for actually planning the implementation of and deploying the knowledge base, the knowledge developers responsible for inputting information into the knowledgebase system, the knowledge coaches who are responsible for ensuring quality of knowledgebase article and improving the skills of the knowledge developers, the knowledge domain experts who are responsible for assessing and addressing gaps are in the knowledgebase, and the managers who support the knowledge process and the people involved (Joslin, 2013). Within the context of SignUpForce, the adoption team

should consist of the Director of Support and select members of the Support and Training teams. The knowledge developers would consist of the entirety of the Support department. For the role of knowledge coach, I would recommend the creation of a new position dedicated to this role or a change in responsibilities for a member of either the Support or Training teams. The responsibilities of the knowledge domain experts would be fulfilled by the Training team and the knowledge coach. The managers involved would be the executives who oversee Support and Training, the Director of Support and the VP of Marketing, although given the new processes and need for improved coordination between the Support and Training departments it may be more efficient for both departments to report to the Director of Support.

Processes for Knowledge Management

Effective knowledge management can be summarized as a series of related activities: searching for and storing knowledge, controlling knowledge, producing knowledge, and updating knowledge (Oztemel & Arslankaya, 2012).

In order for knowledge to be effective and create value for an organization, it needs to be stored somewhere, but also available and useful at the time and place that it is needed; therefore, a search and storage mechanism that is understood by and accessible to all employees must be implemented (the specifics of this mechanism will be discussed elsewhere).

Just as it is important to ensure that the organization's knowledge is where it needs to be at the appropriate time, it is also important to restrict the company's knowledge only to parties that need to know it. For example, if SignUpForce's competitors had the ability to access or even copy the organization's knowledge, the competitive advantage that it could provide would be lost. Therefore, SignUpForce's knowledgebase should only be made available to employees and to licensed customers. Even within this context, there may be situations where even licensed

users do not need to be privy to certain information or system specifications, meaning that the search and storing mechanism needs also to be capable of distinguishing between information that is publicly available and internal-only.

The most time-consuming portion of the knowledge management process is the production and updating of the knowledge itself. The act of documenting knowledge should be primarily driven by individual incidents; in other words, each time a Support representative resolves an issue, that resolution should be documented so that the next time that same issue or question occurs anywhere else within the organization, the solution will be readily available (George et al, 2012). Within this context of incident-driven documentation of knowledge, there are four specific processes that need to be performed: in brief, they are Capture, Structure, Reuse, and Improve (George et al, 2012).

At the core of the Capture portion of the incident management process is the concept of "tacit knowledge" – in essence, the idea that we don't know the extent of our own knowledge until we are asked about it. For example, if an individual is asked to generally write down everything he or she knows about a particular topic, the results are not likely to include answers to many specific questions that could be asked about that topic. This tacit information becomes explicit information when it is recorded in the context of the question in which it was asked for. Simply recording the answer to a question after the question has already been asked and answered is insufficient; the effectiveness of the knowledge is the answer when it is framed by the question (George et al, 2012). By capturing the question that is being asked from the perspective of the customer asking it, the linked answer will be easier to locate by another service representative when asked the same question by another customer. This act of preserving the customer's perspective, even if their perspective is wrong, will make the resultant

knowledgebase article more easily searchable by others. A specific scenario in which this occurs within SignUpForce is in the support of its web-based and mobile-based platforms. There are several components in the mobile platform that draw their configuration data from the web-based platform. When customers report the issue, they frequently perceive and report it as a mobile-based issue since that is where they intend to display their content; however, the true nature of their issue actually lies in their web-based configuration. By documenting these questions in terms of a mobile-based question with a web-based answer, even Support personnel who are not yet fully trained in the mobile-based component will be able to quickly find the correct solution to the customer's perceived issue.

The Structure portion of the incident management process is intended to improve the consistency and readability of the content, specifically with three groups in mind: the customer, the support representatives, and the company (George et al, 2012). The customer's perspective that needs to be captured is of the incident that is being documented, the support representative's perspective that needs to be captured is of the root cause and the resolution, and the company's perspective that needs to be captured is of metadata such as the number of times the support article has been viewed and its modification history. These needs as well as the need for consistency can be satisfied if the support representatives who are documenting the incident do so by separating the information into three straightforward categories: the issue (what the customer is trying to do or what facet of the system is experiencing an error), the environment (what products the customer is using, the type of browser is being used to access the system, and what specific configurations might be employed within the product), and the resolution (the answer to the customer's question or the steps that are required in order to resolve their issue). When the information is presented in this consistent issue/environment/resolution format, its

readability will also be improved (George et al, 2012). The metadata portion of the article should be generated and maintained automatically by the knowledgebase tool, thus freeing the support representative of the need to research and capture this information.

The Reuse portion of the incident management process is intended to both put the information gained and the structure under which it is submitted to effective use during the incident management process. Most directly, if a support representative captures the customer's issue and environment early in a support call, the representative now has enough information to perform an effective query in the knowledgebase in order to discover a solution if one exists or begin writing a new knowledgebase article if one does not (George et al, 2012). Having one centralized knowledgebase allows for this querying to be possible; potential solutions for issues could currently be searched in the SalesForce case notes, the FogBugz bug repository, the help documents, and the user community, and it is not practical for a support representative to have to search four different databases for a solution while on a call. If a resolution already exists, then the customer's incident can be immediately linked to it, further creating metadata for upper management to analyze. Even if a resolution does not exist, similar articles that do not directly provide a resolution but are related may provide additional perspective or questions to ask that narrow the search parameters (George et al, 2012).

The Improve portion of the incident management process is intended to identify which knowledge articles are the most valuable and improve them according to their relevancy to the customer. The cornerstone of this process is the idea that "reuse is review" (George et al, 2012). Each time a knowledgebase article is accessed by a support representative, that support representative assumes responsibility for its accuracy; in other words, if an article is discovered to be inaccurate, incomplete, or have a solution that is inconsistent with its problem statement,

the support representative who is reading it should be expected to correct any errors or update its content with new information. In this way, the goal of collective ownership of knowledge is improved by all involved individuals making constant contributions to the sum of the group's knowledge (George et al, 2012).

Tools for Knowledge Management

With these criteria in mind, the most appropriate tool that satisfies the requirements detailed above would be the SalesForce Service Cloud. SignUpForce currently employs the SalesForce Sales Cloud component for its Sales and Marketing teams; the Support team also uses this component for purposes of case tracking and customer support. The Service Cloud integrates with the Sales Cloud and will allow Support personnel the same access to customer information and history. However, the Service Cloud includes a knowledgebase capability that is integrated with its case tracking (SalesForce, 2014). Additionally, it includes a customer community feature, which would more easily integrate with the Sales Cloud and Service Cloud than the current user community provider, GetSatisfaction. This combination of additional features makes it an ideal choice to expand the company's existing infrastructure.

The Cost of Support and Knowledge Management

Based on cost factors for the Support and Training departments, we can calculate the total cost of SignUpForce providing technical support and training to its users, as well as the total cost per incident. In brief, the cost per incident by communication channel is determined by calculating the total annual cost of operating the Support department and dividing that figure by the number of total incidents with customers that the department makes.

Specific to SignUpForce, the raw costs involved in maintaining the Support department can be estimated as follows:

- Salaries and Benefits: Director (1 at $100,000), Manager (1 at $75,000), Leads (2 at $55,000), Support Representatives (9 at $50,000), Knowledge Manager ($55,000)
- Work Area: est. $20 per square foot per month (est. 20 square feet per seat, or $400 per month per seat)
- Utilities: phone/internet est. $100 per month per seat (14 seats)
- SalesForce Sales and Service Clouds Licenses: $425 per month per license (14 licenses)
- GoToMeeting/GoToAssist Licenses: $55 per month per license (6 licenses)
- Supplies/Coffee/Miscellaneous: $150 per month

When determining the cost per incident, we would ordinarily break down the cost in terms of how a customer contacts the support center (Last, 2005). In an ideal situation, all customer incidents would be resolved on the first contact; if this were the case, it would be a simpler matter to determine how many incidents were resolved by each of SignUpForce's different contact vectors (email, telephone, or user community). The cost factors are slightly different depending on the resources used by each vector even though all vectors require the direct time and attention of a support representative; that is to say, cases resolved entirely by email do not have an impact on the office's telephone bill which may reflect that emailed issues are cheaper to resolve. However, the majority of incidents are not resolved on the first contact, and the management of many of these incidents will frequently vary from the vector of original contact (for example, support representative who initially receives an email from a customer may call the customer directly to get more specific information about the issue; in this instance, both email and telephone resources are being used). Furthermore, the user community cannot be used as an effective gauge for case deflection since the user community does not include a method by which to report on the number of times that a given user community post has been viewed; without

accurate reporting, we can only safely assume that each solution on the user community has only been consumed by the customer who prompted it and thus count the cost of incidents resolved by the user community with the same weight as incidents resolved by telephone and email support.

The Benefits of Knowledge-Based Support

If the knowledge-based support workflows described in previous sections were implemented at SignUpForce, customers would have a consistent, reliable self-help option that could be tracked and factored into the Support team's overall performance. The Consortium for Service Innovation identifies a number of quantifiable benefits as a result of a successful conversion to knowledge-based support; among these benefits are 50% improved time to resolution due to support representatives being able to more quickly find solutions to known issues and resolve issues on the first contact (George et al, 2012) and up to 85% case deflection due to customers having a comprehensive self-help option (CSI, 2007). The full measure of these benefits can be expected to be in effect as soon as 18 months after the knowledge-based support implementation has begun; incremental improvements will be shown along the way as the new process is adopted by both employees and customers (CSI, 2007). If SignUpForce were to begin implementing the knowledge-based support process now, in late 2014, then the full effects would be deliverable in early 2016.

In 2013, SignUpForce's Support department responded to 25,306 incidents through email, telephone, and the user community. With the estimated cost figures presented above totaling $940,985 and dividing our number of incidents in a year by the cost of solving those incidents per year (Last, 2005), we can determine that the cost per incident from all established contact vectors is $37.18. In contrast, the cost per incident when a customer consults the knowledgebase does not require the salary of support representatives or the maintenance costs for the Support

department's physical space and tools to be accounted for since these incidents do not require direct intervention; in effect, the only cost for providing self-help solutions is the monthly per-seat fee for the SalesForce Service Cloud and the salary of the knowledge coach. If 25,306 incidents were answered via knowledge-based self-help options, the total cost of answering those incidents would have been $66,300 with a cost per contact of only $4.82.

The 85% case deflection that was previously mentioned as the goal and expected return for self-service use means that 85% of customers will attempt to use self-service options rather than first opening an incident with Support; of these number of self-service attempts, 85% will find what they need – in other words, 15% will still open an incident even after consulting self-help options (CSI, 2007). Had the knowledge-based support process been fully in place during the 2013 year that had 25,306 incidents that required direct involvement, there would have been 21,510 attempts to address issues with self-help options, with 18,284 of these being successful, leaving a total of only 7,022 incidents requiring direct contact with a support representative. These figures assume that the goal of full adoption and acceptance of the self-help options is achieved; a more conservative estimate might be that 50% of customers will attempt to use self-help options. In this case, there would have been 12,653 attempts to address issues with self-help options with 10,755 of these being successful, leaving 14,551 requiring direct contact with a support representative.

This data can be interpreted and acted upon in a few different ways, one way being that the 7,022 to 14,551 incidents requiring direct responses could have been handled with only 2.5 to 5.2 support representatives, prompting some dramatic reductions in the overhead cost of providing support.

Another way to act upon the data would be that with knowledge-based support in place, the company would be able to grow its customer base without a proportional increase in the cost of providing support. If the current Support team is able to process 25,306 incidents requiring email or telephone contact and there are between 135% and 260% as many more knowledgebase-deflected incidents than there are direct contact incidents, then the knowledgebase would be able to respond to an additional 34,163 to 65,796 incidents per year, for a total possible range of 59,442 to 91,102 incidents answered.

SignUpForce currently has approximately 700 unique users who could potentially generate contacts with Support. Dividing the 2013 incident total by these 700 users shows that each user contacted Support an average of 36 times during the course of the year. Under the assumption that 36 self-service and/or direct contacts per user per year remains constant, then it is safe to assume that if 59,442 to 91,103 self-service and/or direct contacts per year means that the company will be able to support approximately 1,650 (conservatively) to 2,500 (in a best-case scenario) unique users before needing to increase the personnel in the Support department. Although there are frequently multiple user logins associated with each contract, this roughly corresponds to the company being able to increase its revenue and customer base by up to 350% by 2016 without also having to also increase the cost of Support. Without the implementation of knowledge-based support, these increases in the size of the customer base would necessitate the addition of 12 to 23 additional personnel with an additional cost between $678,300 and $1,300,075 per year.

The Scrum Development Process at SignUpForce

The development process at SignUpForce follows the Scrum model of agile development. The backbones of the Scrum process are the user stories, the product backlog, and

the sprint. User stories are requests or ideas from all interested parties about features, fixes, or functions that might improve the software, the product backlog is a collection of user stories that have not yet been developed and implemented, and the development sprint is a fixed period of product development which at SignUpForce consists of approximately one week of pre-planning and design, four weeks of development, and one week of final testing and implementation (Schwaber & Beedle, 2002). The development team performs two sprints per quarter, with the first being primarily related to infrastructure and bug fixes and the second being primarily related to new feature development. The exact duration of the sprint and the work that it will accomplish are determined by the Project Manager. The Project Manager communicates with Product Owners, the individuals who in turn communicate with various internal and external stakeholders in order to establish which features or fixes are the most important within the context of the user base's current needs and expectations. Based on the Product Owners' priorities, the development team estimates the time that will be needed for each priority item and the Project Manager loads the sprint with tasks from the product backlog (Schwaber & Beedle, 2002).

The Role of Documentation in Scrum

As an agile development process that promotes collaboration and communication between stakeholders, Scrum de-emphasizes the role of documentation as the primary method of sharing knowledge between developers (Chau, Maurer, & Melnik, 2003). In traditional development processes, models and formal documentation exist to clearly record the requirements for the project, to ensure that continuity is maintained between the long analysis, design, and implementation phases of development, and to provide stakeholders with an evolving image of what the finished system will look like when it is completed (Rubin & Rubin, 2011). In

contrast, Scrum records its requirements in the form of the user stories in the product backlog, ensures continuity between requirements and implementation through communication between stakeholders, and presents incrementally-improving working versions of the system as opposed to models at the end of each sprint.

Although documentation is de-emphasized in Scrum, this method is not entirely devoid of document artifacts and some of the constructs of traditional development do survive the fast pace and undocumented conversational nature of agile development. The business requirements and functional requirements that would be clearly enumerated in a traditional development model's documents are still present, albeit in different formats; the business requirements for individual features exist in the form of user stories, while the functional requirements are determined during collaborations between developers. The evolving reality of the system is not illustrated through product documentation as in traditional development, but rather is shown through an instance of working source code.

In effect, the document artifacts that Scrum are then most concerned with are the user stories and the source code itself (Cohn, Sim, & Lee, 2009). Were development occurring in a vacuum, then this may be sufficient material serve as documentation for the software product. However, software documentation is rarely written for the benefit of the developers writing the software, it is written for the benefit of those who use, support, and maintain the software (Selic, 2009). Source code artifacts will not be understood by stakeholders who are not programmers, which includes not only customers or third-party partners, but also the internal Support, Training, and Sales departments who ultimately have to digest this material and deliver it to the customer (Rubin & Rubin, 2011). The collaboration and communication that is inherent to Scrum means that individual developers will achieve a high degree of personal understanding of the aspects of

the system and the source code that they work on; furthermore, individuals within the development team and the development team as a whole will retain a high degree of tacit knowledge concerning the underlying workings of the system. Since the collaboration and communication between developers is informal and may not be recorded outside of the source code, this aspect of the system's development remains tacit knowledge accessible only to the developers who participated in these specific collaborations. Another downside to this arrangement is that without this knowledge being recorded, the knowledge leaves the organization with the individual developer if that developer parts ways with the company (Rubin & Rubin, 2011). There are several major components of SignUpForce's systems that were written by developers who are no longer with the company; as a result, these components do not necessarily get the continued development that they need since the current development roster does not have the time to reverse-engineer them based on only the working source code if they are also expected to keep up with new feature development in other areas.

Agile Knowledge Sharing

Although the collaborative development process of Scrum is effective in terms of quickly producing software that reflects the changing needs of the customer base, an unintended consequence of this informal collaboration is a failure to make the development team's tacit and internal knowledge of the system explicit and external for use by other departments and for communication to the customer base. When the production of business documentation and knowledge resources are perceived as processes that are separate from the work processes that are directly related to producing the source code, they will be de-prioritized by management and developers in order to focus on the more important deadlines that directly relate to the release of the software (Levy & Hazzan, 2009).

Within the development organization, the primary goal of knowledge management is to transform tacit knowledge to explicit knowledge and to transfer information from individuals to groups. Attempting to comprehensively document every piece of tacit information in order to blindly fill a manual with information may not be an effective use of development time (Levy & Hazzan, 2009), especially considering that the software that is being documented is constantly in flux due to the changing reality of the Scrum environment and it could very well become outdated while it is still being written. Fortunately, an agile development model maintains other values that are in line with the knowledge management practices that were previously discussed, specifically the emphasis on collaboration and on the collective ownership of a project. This is not to say that there are not impediments to creating an organizational culture that is open to improving knowledge management practices; for example, individual developers may feel that their continued advancement is dependent upon the expertise that they bring to the table rather than the degree to which that expertise is shared, or individuals might not realize that tacit knowledge they take for granted might be beneficial to others in the organization (Levy & Hazzan, 2009).

The development team currently produces release notes that accompany new product features that come out of each development sprint. These release notes are delivered to the Training department. The Training department revises the release notes to be more suitably readable on a less technical, more publicly consumable level. After this revision, the Training department sends the release notes to Marketing, who revises them further and includes them on the public-facing corporate site to coincide with the update to production; this public display of the release notes is not subject to queries to the current incarnation of the user community, nor would it be subject to queries to the knowledgebase when it is implemented. At some future

point not necessarily coinciding with the product update, the Training team will also revise the release notes for inclusion as a how-to article in the help documents found on the support site. During the week prior to the product update, the Training department will conduct training sessions to bring Sales and Support up to speed on how to use the new product features, and present the same information to customers by offering a training course to cover it. This handling of release notes is problematic to Support and to the end user for several reasons. While it is acceptable for Marketing to "sanitize" the information in the release notes to make it more appropriate in terms of selling and advertising the product, the data loss that occurs when the information is communicated from Development to Training to Marketing to potential customers makes this an unsuitable vector for conveying the information from a usability and support perspective. Furthermore, current customers aren't going to go to the public-facing website for support and answers about new features, they are going to go to the login-protected support site where the help documents and user community reside. The fact that updates to the help documentation do not always occur immediately coinciding with the release means that the new information is not immediately available to customers without incurring a call to Support or attending the training course.

The challenge in creating documentation that justifies its cost is to ensure that the process that produces it is not intrusive to the lightweight development process. Development is already producing release notes and Training is already maintaining a suite of help documents, so the most value can be derived from the processes that are already in place by improving the quality of the existing artifacts that are being produced and improving the underlying culture of communication that binds the organization together.

Improving the quality of the release notes during the development process can be done by linking the release notes for any given user story to the item that developers are working on to ensure that the notes are written as the code is being generated and tested by the developer. When the feature is completed by the developer and sent to QA for testing, the QA analyst who works on it needs to not only validate the performance of the new function against the original user story, but also validate the performance of the new function against the documented behavior (ISO, 2012). By generating the release notes during development and validating them during QA, the act of documentation does not become a separate chore that someone has to perform after the fact and the process becomes non-intrusive. By keeping the documented release notes focused on the behavior of an individual feature from the perspective of how it will be used by the customer, then it avoids the needless complexity of code-level documentation and improves the quality of the information that is being delivered to Training to communicate to the customer.

Apart from the standardization of the information that is being externalized, there is also room to improve the method of delivery to the customer. The Marketing department can do what they will with this material to advertise the product to potential customers, but this should not preclude the distribution of release notes on the Support side as well, potentially as a subsection of the help documents indexed by date. Ultimately, the existing help documents do serve effectively as a user-friendly guide on how to navigate the different features of the software. The principle issue with the presentation of the help documents is that they are presented as tacit knowledge, while customers are going to need information from them as explicit knowledge. The help documents are displayed as a list of eight sections; each of these sections can be expanded to show links to individual articles that explain how to use specific

product features. If users have a specific question about how to use something, then they have to browse through eight categories worth of article titles in order to find the answer to a question; the help documents can be queried, but the search bar that allows this is at the bottom of the page and isn't immediately apparent until the table of contents has already been browsed through. Moving the search bar more prominently to the top of the page would significantly increase the utility of the help documents. In addition, links to the help documents should be included as entries in the knowledgebase so that they can be discovered as results to queries in that database as well, framing the search terms that locate them in terms of the questions that users might ask about how to use particular features.

Divisions in the Organization

Under the company's current organizational structure, there is relatively little interaction between departments; when executive and director-level employees meet, they will pass a brief summary of these meetings on to their subordinates, but by and large the rank and file in different departments do not have any need to communicate during the daily execution of their responsibilities. The only tangible benefit produced by this sharp division of labor is that developers are less likely to be bothered with external requests while they are otherwise engaged in a sprint; otherwise, the lack of cross-departmental communication results in a collection of corporate sub-cultures evolving and potentially diverging rather than a single corporate culture united by the company's common values. With employees focused so narrowly on their departments' specific responsibilities, it is more likely for departments to form rivalries with one another and for employees to lose sight of the bigger picture of the company's mission: providing excellent service and an excellent product to the customer (Maturi, 2013).

Cross-Training to Promote Knowledge-Sharing in the Organization

Cross-training employees between departments would help to promote a culture of knowledge-sharing across the organization as well as provide other intangible benefits to the organization. Cross-training employees can benefit the department that is doing the training; a trainee can frequently bring a fresh perspective, new insights, and original solutions to old problems or inefficient processes. Employees who have been trained in the work activities of departments other than their own will be better informed about the processes involved in other components of the company. Being aware of the workings of all parts of the company will make the knowledge that is specific to the technical functions of the product more relevant in the context of the entire organization's workings and provide context for otherwise arcane development processes. With awareness and education also comes a certain degree of empathy; understanding what a colleague puts into their job and how their efforts benefit the overall performance of the company can result in a greater degree of cooperation between employees from different departments (Maturi, 2013). Cross-training also embraces the agile development values of collaboration and collective ownership that have been discussed elsewhere in this paper, but applies these values across the entire organization instead of just the source code or the knowledgebase. With all employees armed with the experience of being engaged in more parts of the company's services and processes, employees will be more energized in their normal work and more personally invested in the company's success (Maturi, 2013).

For most of its fifteen-year history, SignUpForce has been a small company with fewer than thirty employees. Any personnel issues or employee dissatisfaction that occurred became quickly apparent to management and easy to remedy or remove. A handful of surveys have been conducted to objectively measure employee satisfaction, but these have been collected with sufficiently erratic frequency as to render any metrics that could be derived from them irrelevant.

Likewise, a matrix for gauging employees' level of skill in different areas does not currently exist within the organization. As a result, the benefits to employee knowledge and satisfaction that cross-training might produce will be difficult to measure against the cost of implementing them.

The goal of cross-training at SignUpForce is not to create a completely interchangeable workforce. In theory, cross-training could be applied to the point where any individual in the company could fill any role in an emergency, however there exist certain skillsets such as code-level development or IT infrastructure and certain levels of permissions such as information security, human resources, or database administration that should not be casually handed out to everyone. For the purposes of promoting knowledge-sharing in the organizational culture, a basic competency in and understanding of key inter-departmental functions is all that is necessary. Cross-training assignments should be strategic rather than universal in order to train employees in the departments where they will learn the most effectively. The major organizational divisions that comprise the bulk of the company's personnel are Support (13 representatives, leads, and directors), Development (19 developers, QA analysis, and project managers), and Sales (13 representatives, leads, and directors). Out of these departments, Development rarely interacts directly with current or potential customers, Sales interacts almost exclusively with potential customers but very little with current customers, and Support interacts exclusively with current customers within the context of their use of the product.

The members of each department would receive different benefits from cross-training outside their own experience. Developers and QA analysts would receive the most effective cross-training stepping into the role of Support. Developers have expert knowledge regarding the code-level functions and technical implementation of the system, but they do not have a

widespread understanding of many of the system's functions from the view of how a customer actually uses them. The Scrum model is intended to remove obstacles to communication between the customer and the developer in order to ensure that the needs of the customer are clearly expressed to the developer who will be fulfilling them, but this aspect of the development model is not practiced to an effective degree. A cross-training term in Support will not revolve around any specific feature or function, but will serve to provide developers and QA analysts with a more general perspective of and empathy for the user's experience. Sales representatives would receive the most effective cross-training stepping into the role of Support; by talking with current customers and assisting them with their use of the software services, they will increase their own knowledge of the software's capabilities and limitations as well as gain a better understanding of how customers use the software. With a better understanding of how customers use the software that they have been sold, sales representatives will be better able to target sales to the needs of prospective customers. Additionally, during their cross-training experience, sales representatives will receive more hands-on experience with the knowledgebase which will increase its use as a resource outside of Support. Support representatives would receive the most effective cross-training stepping into the role of QA. By spending time working on escalated and technical issues and not having to directly manage customer expectations, support representatives will be able to improve their troubleshooting skills. Since the QA role is more directly engaged with the software development process than the support role, support representatives will also gain a first-hand understanding for how the software development process functions. A cross-training assignment as a QA analyst is more appropriate for a support representative than a cross-training assignment as a developer; support representatives are already expected to perform some troubleshooting as part of their normal responsibilities

Since a cross-training initiative pulls skilled employees away from their usual duties and places them into the role of a potentially less productive trainee in another department, there is a direct, measurable loss of work output as a cost for the immeasurable gains previously stated. Based on the number of current personnel from departments that are being cross-trained, if each employee spends one week per year cross-training then the company as a whole will experience 45 person-weeks of lost productivity. Estimating an average salary and benefits of $75,000 per employee these 45 weeks spent cross-training would result in a total cost of approximately $65,000 per year (in effect, the cost of hiring an additional employee). Different departments will also face specific challenges to implementing a cross-training program. The Support department is staffed according to schedules that satisfy peak times for call volume; if representatives who cover certain shifts (for example, the very early and very late shifts) are diverted to training as a QA analyst, then schedules need to be rearranged elsewhere in the department in order to ensure that these shifts are covered. Since calls into support spike during the week immediately after a development sprint's implementation in production, cross-training should not be planned during these weeks. Similarly, the Development team will not want to divert any personnel away from the weeks of a sprint where planning and implementation occur; with eight sprints occurring per year, it is safe to assume that one or two developers will be absent and cross-training at some point during the four weeks of development time, resulting in a loss of 40-80 hours of developing time from what the team as a whole will be able to accomplish in a given sprint.

The implementation of a cross-training program includes some risk to its success and positive impact on the organization. The largest risk is a lack of buy-in from management and employees. If executives and director-level employees are not committed to cross-training and

improving communication, camaraderie, and knowledge-sharing between departments, then there is no reason to expect that lower-tier employees will be committed to it either. If employees merely pay lip service to their time in another department, then the productivity that is lost as a result of them being away from their normal responsibilities will truly have been wasted. In order for employees of any tier to be able to fully immerse in and commit to their temporary reassignment, the employees who are not cross-training must be able to function without them and cover for their absence as though they were on vacation or otherwise unavailable for work; the departments at SignUpForce that have been singled out for cross-training have sufficient redundancy to be able to survive without any given employee for a week. For a cross-training experience for an individual to be successful, that individual must actually fill the role and perform the functions of the job that is being trained as opposed to simply shadowing another employee.

Conclusion

In conclusion, SignUpForce is in a strong position to reduce its operating costs for Support and increase its profitability as it grows. The implementation of a tested and proven knowledge-based support methodology will significantly change the way that the organization interacts with its customers and improve the quality of its service. The implementation of a cross-training program will improve communication within the company, develop empathy between different departments, and increase the focus that the company has on its customer.

The lesson to be learned from this project is that a social, collaborative software development process like Scrum should remain social and collaborative without trying to impose the arbitrary regimentation of comprehensive documentation into its proven model. With that in

mind, there may still be room for future research into processes that formalize some aspects of documentation within the context of an agile development environment.

Resources

Chau, T., Maurer, F., & Melnik, G. (2003). Knowledge Sharing: Agile methods vs. Tayloristic Methods. *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003.* WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on pp.302,307, 9-11 June 2003 doi: 10.1109/ENABL.2003.123427

Cohn, M., Sim, S., & Lee, C.P. (2009). What Counts as Software Process? Negotiating the Boundary of Software Work Through Artifacts and Conversation. *Computer Supported Cooperative Work: The Journal Of Collaborative Computing*, 18(5/6), 4011-443. doi: 10.1007/s10606-009-9100-4

Consortium for Service Innovation/CSI (2007). KCS Measurement Matters: The Benefits and Organizational Measures of Knowledge-Centered Support. Retrieved from www.serviceinnovation.org/included/docs/kcs_benefitsandmeasure.pdf

Coulson-Thomas, C. (2013). Knowledge management and the high performance organization. *Management Services*, 57(4), 41-47.

George, M., Kay, D., Oxton, G., Joslin, R., MacIntosh, J., & Murray, K. (2012). Knowledge-Centered Support Practices Guide. Retrieved from http://www.serviceinnovation.org/included/docs/kcs_practicesguide.pdf

Joslin, R. (2013). Investing in Knowledge-Centered Support: Knowledge Management Best Practices Within Service Management. Retrieved from http://www.thinkhdi.com/~/media/HDICorp/Files/Education/KCS/hdi-invest-in-knowledge-centered-support.pdf

Last, R. (2005, September 14). Understanding Cost per Contact by Communications Channel. Retrieved from

http://www.thinkhdi.com/member/login.aspx?returnurl=/topics/library/books/metrics-guides.aspx

Levy, M., Hazzan, O. (2009). Knowledge Management in Practice: The Case of Agile Software Development. *Cooperative and Human Aspects on Software Engineering, 2009.* Chase '09. ICSE Workshop on, pp.60,65, 17-17 May 2009. Doi:10.1109/CHASE.2009.5071412

Maturi, R. (2013). Cross-Training: Creating and Implementing a Successful Plan. Retrieved from http://www.areadevelopment.com/laborEducation/Q1-2013/implementing-cross-training-hot-back-ups-37372612.shtml

Oztmel, E., & Arslankaya, S. (2012). Enterprise knowledge management model: a knowledge tower. Knowledge & Information Systems, 31 (1), 171-192. Doi: 10.1007/s10115-011-0414-4

Rubin, E., & Rubin, H. (2011). Supporting agile software development through active documentation. *Requirements Engineering.* 16(2), 117-132. Doi:10.1007/s00766-101-0113-9

SalesForce Service Cloud Overview (2014). Retrieved from www.salesforce.com/service-cloud/overview

Schwaber, K., & Beedle, M. (2002). *Agile Software Development with Scrum*. Upper Saddle River, NJ: Prentice Hall.

Selic, B. (2009). Agile Documentation, Anyone? *Software, IEEE*, vol. 26, no. 6, pp.11,12, Nov-Dec. 2009 doi: 10.1109/MS.2009.167

Systems and software engineering – Developing user documentation in an agile environment. *ISO/IEC/IEEE 26515 First edition 2011-12-01; Corrected version 2012-03-15*, pp.1,36, March 15 2012 doi: 10.1109/IEEESTD2012.6170923