Building an Agile Culture in a Regulated Environment

Michael Meissner
Omnyx.com
(A Joint Venture of GE Healthcare and UPMC)
michael.meissner@[ge,omnyx].com

Abstract
Starting with its inception in June of 2008, Omnyx has implemented scrum from ground up, building a very successful agile culture in a regulated environment. We very intentionally shaped the organization, cadence, and office space such that it was most conducive to scrum and to best facilitate an atmosphere of total visibility and transparency. The foundation of this agile culture is based on a set of essential core values, striving for excellence, and continuous team building events to reinforce that trust and healthy conflict are integral to self-organizing teams. While agile and regulated environments often are at odds, we successfully managed to overcome the divide, maintaining the key benefits of agile while accomplishing compliance. With a 2+ years development cycle for our first release, we applied a continuous Alpha/Beta process to ensure we received critical feedback while not yet having an install base. Last but not least, we firmly believe that the #1 priority for any leader advocating agile needs to be fostering a strong foundational culture as well as making sure the right people get on and the wrong people off the bus.

1 Introduction
The agile manifesto speaks to the fact that the goal is to put running software and people interacting with another above processes, documentation, contracts, etc. The chicken and pig reference often used in scrum emphasizes this by making a point that one needs to get together the people who are committed vs. the ones who have thoughts and opinions but are only involved in the process of producing the deliverables. There is plenty of literature that following such a framework can result in higher product quality and higher customer satisfaction with reduced time and cost. At the foundation of this is the principle of self-organizing teams that are capable of taking on incremental slices of work and self-organize the project from inception to completion. While this sounds simple and straightforward, there are two significant core challenges to make this work very well:

1. Self-organizing and high-performing teams
2. Teams to manage all necessary project deliverables

Self-organizing and high-performing teams (refer to 1.) is truly a challenge, even if an organization manages to entirely remove any other organizational impediments (which is hard, too). While known to be the most powerful form of accountability and well surpassing effectiveness of accountability to a superior/manager, peer accountability requires team members to overcome their (i) absence of trust, (ii) fear of conflict, (iii) lack of commitment, (iv) avoidance of accountability, and (v) inattention to results (Lencioni, 2006). Too often teams and individuals overlook the first two aspects (trust/conflict) and directly go on to focus on the technical considerations of delivering the line items committed to within a sprint or release. Over time, this puts strains on a team and eventually limits the team’s productivity and effectiveness. We recognized early on that this is fundamental and established a set of core values, an excellence credo, and organized reoccurring teambuilding events as an integral part of building a strong agile culture.

Teams to manage all necessary project deliverables (refer to 2.) is the other significant challenge because it requires a horizontal mindset from people who are used to be operating in a vertically aligned organization. The classic mechanism to address this is by establishing a “Definition of Done” guideline that defines what needs to be completed before a user story can be considered “done”. Unfortunately, regulated environments such as FDA and ISO mandate a set of additional requirements before a (software) system can be considered done. This stems from a more traditional thinking (waterfall) where phase based design controls and design verification as well as design validation are to be completed and very well documented prior to a device being considered shippable. While regulated environments do not mandate a waterfall approach, development efforts in the past decades have been largely dominated by it and thus, the culture is not agile. If you were to read
the agile manifesto to experienced Quality people, it is safe to assume that (s)he will show symptoms of sever goose bumps and probably other more severe symptoms, due to this significant culture difference. We recognized early on that in the process of establishing our own Quality System, governing our development process, we needed to carefully pay attention to the details to ensure (i) we are compliant while at the same time (ii) not mandating a process that would present us from harvesting the benefits of an agile development methodology.

1.1 Project Background
Omnyx is a joint-venture between GE Healthcare and University of Pittsburgh Medical Center (UPMC), developing an enterprise wide Digital Pathology product. To date, routine clinical workflow uses glass slides and paper records associated with those that are distributed within the healthcare enterprise. The company’s mission is to fully digitize pathology such that all work is done electronically, including the digitization and distribution of the image information. For institutions such as UPMC this requires handling of patient data for ~300,000 cases with a total of ~2,000,000 WSI, each ~500MB after compression (~1 peta byte annually), distributed across multiple hospitals in geographic regions stretching across continents.

The company is in the process of bringing its product to market in 2011. When the company was founded in summer of 2008, appropriate funding was provided to build the organization from scratch.

Very early on, we chose Agile (scrum) as the approach to go about developing our product(s). The choice was based on the fact that digital pathology is going to be a new market and the target users are currently doing their routine work using microscopes. Even though we had worked with the Carnegie Mellon University’s Human Centered Design group to design the system in great detail and to ensure usability was built in from the beginning, once stakeholders would interact with a computer instead of a microscope, we expected that we would have to frequently discover need for change rather than following a pre-determined plan.

2 Building the Foundation
We intentionally shaped the organization such that it was most conducive to scrum, providing a fertile ground to build a strong agile culture. In the following we will summarize some of our key accomplishments that we consider critical:

1.) Leadership buy-in:
Fundamental to our scrum roll-out was a strong internal vision and leadership. At the inception, we convinced senior management and got broad internal buy-in from the CEO and VP level. The broad support allowed us to establish a real organizational heartbeat with upper management regularly participating in the sprint and release reviews. Despite having conducted 8+ internal releases and 60+ sprint iterations, we continue to have broad visibility and participation across all teams, including the CEO. During the calendar year of 2009, the consultant spent 2 days a month in our office, providing training to new hires (we tripled our staff during that time period) and helping with advice on how to refine our processes.

2.) Office space:
The space we moved into offered a large degree of flexibility as only few hard-walled structures and pillars existed. As a first step, we made a conscious decision to building out a large scrum room in the center of the space and made all of its walls whiteboard walls. This scrum room became the center of our universe for interactive planning sessions, sprint/release reviews, and retrospective meetings with plenty of space for the teams to put up sticky notes, post user stories, defects, draw up designs as necessary, or write notes. Following that, we established cube space around the central scrum room to minimize the walking distance. Keeping in mind the principles of maximizing interactivity and removing barriers, we aligned cube space in 6-packs. 6 cubes were aligned such that by simply turning around every scrum team member was able to talk to his/her team mates instantly without having to get up and walk around his/her cube or shout over the cube wall(s). While this worked out very well, over time some neighboring 6-pack teams removed tiles from the cube wall material to open direct communication with the neighboring team rather than having to walk around the 6-pack walls; clearly evidence of how essential a the office layout is for good communication.

3.) Rhythm:
Similarly to the office space, we made a conscious decision to put visibility and transparency first, establishing a powerful organizational heartbeat. It consists of 3 month long internal releases divided into 6 consecutive sprints. Sprints start every second and fourth
Tuesday of the month depending how the calendar days map out, there is a 3 week sprint every now and then. The internal sprint reviews are giving each team time to demo their completed user stories (present value), take questions from the audience, and to summarize where they stand with respect to the current release target(s). Each of these reviews is 60-90 minutes long, open to all employees, and typically with a very high attendance, including VP and C-level staff. We regularly get questions from commercial staff which underlines how valuable the visibility and information sharing is to the organization as a whole. External stakeholders are invited to the sprint reviews, too, but only rarely are able to attend due to their busy schedule and the travel that would be required. Our 3 month release reviews are 2+ hour meetings where we invite IT, histology, and pathology stakeholders from one or multiple institutions we collaborate with closely. During the release review, the purpose is to give these stakeholders ample time to interact with the software. After the demo, we follow-up with a panel discussion where the external stakeholders give us feedback as well as rate us with respect to where they think we are. Furthermore, it allows the internal staff to ask questions and bounce ideas around. Last but not least, the demos, discussions, and panel are summarized and entered into our backlog for prioritization in future releases.

Last but not least, during the execution of the current 3-month release, the POs and scrum team representatives perform high-level planning and refinement of the backlog for the subsequent release which is matured throughout the current release and finalized in its last sprint. Early on, we ran into issues that the transition from one release to the next was exhausting as all the planning came together at once and interleaving the release planning has helped improve this but is not yet ideal.

4.) Matrix organization:
We put great emphasis forming a matrix organization where the staff of each scrum team reports not into one manager but into a number of different managers. This ensures that scrum teams can freely take ownership while turning managers into coaches who have the challenge needing to keep track of the individuals across multiple scrum teams. Please note that the staff consists of software engineers, software testers, system engineers, build engineers, technical writers, etc. so only very few departmental barriers exist. Most organizations do not have that luxury as these type or roles typically do report into different groups or departments.

5.) Implementation and Support:
Similarly, we hired staff for the implementation and service teams and made sure they equally participate in the sprint and release cycle, uniquely allowing those downstream functions to gather visibility in what is coming in future releases. Vice versa, it also provided great visibility to the engineering and product marketing teams on what issues need to be resolved downstream. Several times this visibility was instrumental to understanding what tool and functional gaps existed and we do expect this to be of increasing value to our organization as we role our product out beyond Alpha and Beta sites.

6.) Other parameters:
Each scrum team has a tester and a number of developers as well as a PO. The test to developer ratio is slightly higher than 1:3. We do have a core scrum team which consists of build engineer, packaging engineer, technical writer, automation engineer, and Quality project manager who provide services to the other scrum teams on an as needed basis. Our Definition of Done (DoD) demands that defects related to a new user story are being fixed within the ongoing sprint, prior to acceptance of the story. User story related documentation updates (software design documents, manuals, packaging, etc.) are also part of the acceptance criteria for the user story. Our unit test code coverage rate has been in the 70-80% range and for the past 6+ months we have put stronger emphasis on test automation to allow for regression defects to be found earlier and with less manual labor. We use continuous integration with unit test and some test automation harness in combination with a nightly build that runs a more extensive deployment test on a set of configurations, i.e. a multi-server deployment scenario.

3 Key Challenges
While we were very successful building the substantial foundation, as outlined in the previous section, we did encounter a number of more significant challenges we would like specifically share with the reader of this paper:

1.) Self-organizing teams:
The biggest gap we found in self-organizing teams is the challenge for people on a scrum
team to constructively challenge another. Building foundational trust between scrum team members so that healthy conflict can occur is unfortunately not something we learn during our education as software engineers and is hard to learn or overcome. While critically important, it is often neglected or overlooked and too often solely the amount of new code or features being delivered seem to matter. Building an organization fairly rapid and from scratch did not help us on that front. Every 3 months, we would add another scrum team, rapidly growing and putting new stress on the organization that didn’t have time to settle. There was a continuous atmosphere of change and we ran into several situations where team members would be “venting” about their disagreements with peers to their managers but not bring the same issue up in the scrum team. Without a doubt, this is a tough problem and does require active coaching as it typically does not solve itself.

2.) Skillset gap:
Another challenge of ours was that not too many people out there seem do have a good agile background or experience. Even today, most people come from traditional waterfall shops or watered down agile scrum where a daily standup meeting three times a week is the most agility they ever adopted. Furthermore, it is even harder to find any people familiar with developing software in regulated environments, a key ingredient for us. The main issues is that people who have not operated in such an environment seem to quickly dismiss the need for generating documentation such as code review artifacts, requirement, specification, and design documents, etc. To file the product in compliance with FDA and ISO requirements, such artifacts are essential. The agile manifestos states “working software over comprehensive documentation” and it often felt that any type of documentation was considered an annoyance. On the other hand, some of the few experienced Quality people we had demanded 100% code reviews and an overkill of documentation that literally could put your productivity to a halt. We clearly had work to do.

3.) One of our biggest challenges probably was the fact that we were building a complex product for almost 3 years while never having an Install Base running the release product. The Omnyx Integrated Digital Pathology solution covers almost any challenge one can imagine for software and system design: server backend with multiple distributed servers across geographically distributed locations, databases, fault tolerant backend configurations, data redundancy, large scale storage solutions (federated architecture), backup support, high-availability, high-performance client workstations that support image streaming with low network utilization, web clients for consultations, security, user authentication, user roles, complying with HIPAA, connecting with AP LIS (Anatomical Pathology lab information systems), supporting DICOM, etc. Building such a system literally from scratch and not having a customer base to incrementally work with from release to release was truly a challenge. We had to come up with a way to include active user feedback into the development cycle.

4.) Last but not least, we continued to struggle with the concept of dedicated scrum masters. Organizations typically do have budget constraints and we were to exception to that. Having dedicated scrum masters felt like a luxury that was unnecessary, especially since we did not have the classical obstacles most established companies have that can keep a scrum master busy. Instead, we decided to get more developers and leaders on the team(s).

4 Addressing our Key Challenges

4.1 Core Values and Teambuilding
We started out by introducing the concept of The 5 Dysfunctions of a Team (Lencioni, 2006). Leading the teams through a number of exercises as well as per scrum team assessments, this provided an initial reference point to understand their strength and weaknesses, primarily focusing on building trust and overcoming the fear of conflict. On almost all teams, there were disagreements about some of the resulting ratings, even though the assessments were filled out by the team members and nobody else. Over time, we complemented this assessment with MBTI (MBTI) as well as TKI (TKI) assessments, enabling the team members to understand the interplay of personality and conflict profiles that are present on the teams. Not one but rather the combination of all assessments helped to better understand and recognize some of the team dynamics present. For every sprint, we would try to organize one team building session during which teams were mixed up across scrum teams to master a given challenge. The goal of the challenges typically was to exercise role play and to find a solution while being under some sort of stress (time limit, competition, etc.).
These team building events fulfilled multiple purposes: (i) people got to know each other better, (ii) there was learning in what went well or could have gotten better, and (iii) it provided insights into coaching opportunities to work with individuals and/or teams. While it is hard to pinpoint any single productivity improvement to any one team building exercise, we did complaints when team building events did get cancelled and over time there was an appreciation of the culture and the fun environment they introduced. One of our team building events was for the teams to come up with a set of Core Values. We subsequently refined those with the teams and then adopted them which included making it a habit to start every biweekly team meeting with the question “Any nominations for who went above and beyond exercising our core values in the past 2 weeks?” as well as making the core values part of the performance appraisal process. We believe that the sum of these efforts made a significant difference.

After working with the teams for almost 2 years, the teams came back and asked for help with providing peer feedback as it is quite difficult to do. We then introduced the Situation, Behavior, Impact (SBI) technique from the Center of Creative Leadership (CCL) and are currently still refining this. It was positive to see that the teams took it to heart that this is an issue they need to address and consequently asked for help.

While all of these team building efforts are clearly valuable, it is important to note that those are not one-time investments but a continuous investment we made and continue to make. Similar to requiring a number of harnesses for a high quality code base, one needs to put in place a number of mechanisms to build a strong agile culture and teams.

4.2 Regulatory Environment

Companies selling a product (typically referred to as medical device) requires them to adhere to US FDA Good Manufacturing Practices requirements (FDA, 2010) for the US market or to ISO 13495 (ISO, 2003) for Europe. Other markets typically use the same or very similar requirements. The essence of those is that a Quality System needs to be in place that documents the controlled practices a company uses to define, design, develop and manufacture a device. Having to establish a Quality System from scratch was a key advantage as it allowed us to shape the Design Controls and other process such that those were not creating any major obstacles for our agile culture.

Furthermore, the regulations state that the manufacturer shall establish and maintain procedures to control the design of the device in order to ensure that specified design requirements are met. Typically, this requires establishing high-level product requirement definitions (PRDs) as well as derived software requirement specifications (SRSs). Design Verification ensures that the software requirement specifications are being met and Design Validation ensures that the product fulfills its intended purpose. Last but not least, one needs to establish full traceability from PRD to SRS to Design Verification and Design Validation. What we did in order to establish and maintain this information in RALLY, we had to 1st validate the tool (RALLY) and then implemented the following hierarchy:

- Product marketing folks create product requirements in Rhyma FeaturePlan and push them into the RALLY workspace.
- As part of the release planning, scrum teams translate the PRD stories into one or multiple SRS stories that are children of the PRD story (which could be refined in sprint planning, if needed).
- We introduced a custom field to user stories to denote PRD, SRS, and other user stories as well as what was in/out of a release.
- An SRS level story too big to take on in one sprint and would end up being split into multiple smaller chunks, resulting in a 3rd level of user stories in the hierarchy as depicted in Figure 1,

In combination with the above mechanisms, we developed a set of tools that allow us to extract all PRDs either per project or entire workspace and export them into one or multiple PRD documents. We did the same for SRS documents, keeping a unique identifier for each that is used in the design traceability which is also generated automatically. While this works well for capturing the state at a given point in time, the biggest challenges with agile tools such as RALLY or others is that anybody could make any change any time and thus alter the device documentation. Since it is currently impossible to prevent this, we put mechanisms in place to catch such scenarios by creating e-
Mail notifications, checks as well as diff tools to make sure we were able to ensure the integrity of the data from release to release. All exported documents are checked into our Part 1 compliant DHF tool and approved with electronic signatures. While we do have this well under control, there are a number of details that at times can make this a fairly labor intense process. E.g. the failure mode and effects analysis as well as the risk management process need to refer to individual specific requirements which we currently flag manually. Once we do have our first product released, we plan to use one or multiple Engineering Change Requests to control the change sets being introduced per 3 month release. Thus, the manual flagging and cross-checking will be manageable. For patches to resolve urgent field issues, a faster turn-around time will be used but the change set in those is fairly minimal and thus again manageable. Having said that, agile tools are currently (too) far away to support what is needed in regulated environments. The tools will need to evolve such that these artifacts can be more easily exported and controls be put in place to protect already completed / approved features, test cases, etc. The current gap is a true burden and requires us to have a full time position dedicated to manage those scripts and the integrity of the data. This is likely impractical for other organizations and they are likely to fall back to more rigorous tool sets which don’t integrate well with agile practices. Some major advancements are needed in the area of these tools, especially around configuration and change set management.

4.3 Continuous Alpha/Beta
To overcome the challenge of not having product in the field and being in internal development for almost 3 years, we did continuously make heavy use of our release review meetings with external stakeholders. In addition, our HCD staff and others would meet with customers to get early prototype feedback from multiple sites so that the design could be vetted as much as possible. However, most valuable is when users have continuous access to software so that they can repeatedly use it. Therefore, we engaged a number of sites in a continuous Alpha/Beta process (ongoing since 15+ months) in which our Implementation and Support team would provide an update every release and dry-run validation and usability tests to collect feedback and feed that back into the development process. The amount of changes coming from this was quite significant and one of the most valuable exercises we went through. Whether this was sufficient or not is something we will only find out in late 2011 / early 2012 after the product has been in the field for a few quarters. What we do know is that without it we would have encountered a lot of issues in the field instead of catching them when we did: earlier / before going to market.

4.4 Scrum Master
As mentioned in the previous section, we did decide to not fund dedicated scrum master positions. We felt that we did not have many of the obstacles typically present to keep a scrum master fully occupied, even if shared across i.e. 2 teams. Instead, we decided to assign one
person per scrum team to be a scrum team representative. That person would attend the scrum of scrum meeting (S2S) to communicate up as well as back to his/her scrum team what was decided. If any organizational obstacles were raised, those would get escalated at the S2S meeting so that the managers and the project manager in charge of the compliance artifacts could take care of those. While this worked generally well for us, we do strongly believe that some of the early issues and gaps around agile culture was due to the fact that we did not have sufficient bandwidth to provide leadership and guidance. We did recover from it but daily guidance beyond just removing obstacles is critically important early on and a key function a scrum master could provide.

For some time, we had a dedicated scrum coach who would work with any team or individual that needed it. One challenge was that the coach’s expertise was request driven rather than somebody being continuously observing. In addition, the coach was not able to dive deep into the (technical) issue the scrum teams were wrestling with and thus, the value of the feedback provided was limited. Over time, this eroded trust and credibility and we discontinued the approach. Without any doubt, we do believe that having an internal coach could be a very valuable role but with strong teams in place, it requires a certain caliber of person to add value and work well with the teams.

5 Lessons Learned

After having been in the trenches for the past 2+ years, it seems straight forward to document our approach but by no stretch of the imagination was it trivial to accomplish. All items listed above took collaboration, convincing, coaching, as well as significant effort to put in place. Generally speaking, we would not omit anything what we did but we do have a number of things that we wish we would have done differently / better and would like to pass on to others as advice:

1.) Establishing a strong cultural foundation is essential and while we did spend a lot of effort on it, we would strongly recommend establishing that very early on and to live and breathe the culture continuously and without any compromises. Core values and striving for excellence provide a frame of reference to distinguish what matters and what does not. More rigor, clarity, and facilitation around teambuilding, broken build culture, DoD, pulling-the-pain-forward (quality mantra) etc. could have helped us get to where we are today faster.

These are all key instruments to empowering people. We should have invested more time in exercises to building trust and overcoming fear of conflict (Lencioni, 2006) helping to facilitate difficult but critically important discussions when they need to happen. For a long time did we witness and insufficiently tackle the symptom of people “venting” to their managers or others while not bringing the issues up where they should be: in the respective scrum team exercising peer accountability.

2.) When establishing a Quality System, we should have spent even more time on it rather than letting it take so much time (~2 years). While we had strong internal leadership, engaging an external consultant could have helped overcome some of the obstacles faster, as long as such a person does not only have ISO/FDA experience but also strong software development experience and/or background. Most people in this space do come from a (hardware) device background and tend to apply hardware rules to software. While clearly this is going to be problematic from a practicality as well as productivity perspective, getting external advice can help bridge the gap and overcome the sometime large divide in a shorter time frame.

3.) The use of checklists (Gawande, 2011) and applying them consistently across scrum teams is something that we should have done earlier on. For too long did we fail to provide general guidelines on critical issues such as code quality, defects, user story acceptance, design documents, manuals, broken builds, etc. While the teams should be allowed to self-organize, if every team does go about things differently then it will become quickly confusing to almost everybody, removing the critically important common “frame of reference”. Being reluctant in combination with our lack of scrum master per team did result in a number of debates and wrong turns taken.

4.) Last but not least, make sure to adequately staff your scrum master and managerial coach layer. For the first year, we were too focused on getting horse power quickly on the scrum teams, hiring developers and testers, while not having enough capacity or not being effective enough in providing guidance and coaching on the bigger picture of scrum and regulated environments. This ultimately cost us productivity by having to sort through “process”-issues over time, correcting course on definition of done etc. While we did recognize the problem, it took us
unexpectedly longer than anticipated to finding the right skillset, personality, and attitude. As Agile puts so much emphasis on people, it is critically important to find the right people. Especially for the scrum master / coach as well as managerial positions, it was hard to find the right people with the right passion. Vice versa, it is equally important to remove individuals from the organization that do not fit the culture. With all the literature out there about Agile and scrum, it seems almost like a shortcoming that “Getting the right people on and the wrong people off the bus” (Collins, 2001) is a critically important topic but not mentioned all that often. It should be. It is essential and we don’t seem to talk about it often enough.

6 Conclusion
We presented our approach to forming a strong agile Culture in regulated environments (ISO/FDA). The key ingredients to this were establishing a strong culture based on core values, a culture of striving for excellence, and regularly conducting team building events. Our hiring practices and the mantra for pulling the pain-forward was critical to reinforce the culture and to allow people to focus on code quality rather than just features. As Lou Gerstner, former chairman and CEO of IBM stated: “Culture isn't part of the game – it is the game”. Done well, it can help a great deal to make teams truly self-organizing, one of the biggest challenges in Agile.

For implementing scrum in a regulatory environment, we presented our approach that manages to maintain the key benefits of agile. We accomplished this by getting involved in the creation of the Quality Manual / System early on, providing input to ensure that procedures and work instructions do not hamper agility while maintaining compliance. While it was critically important to work with the Quality team, it was equally important to educate the development teams about the need for adherence to a minimal process and documentation, including appreciation for Quality representatives having to present it during inspections and audits. It is important to recognize this is a two-way dialogue and collaboration.

Last but not least, it is very important to realize that today’s agile tools are insufficient to handle essential artifact generation and maintenance in a compliant fashion. We overcame this by writing our own tools around RALLY and putting in place monitors to check when critical user stories changed containing requirements and specifications critical to the Design History File artifacts. We are actively working with one of the vendors and strongly believe that a broader adoption in regulated environments is possible, if the agile tools would provide basic capabilities needed in that domain.

7 Acknowledgments
We would like to thank Pete Behrens for collaborating with us on this exciting project and for providing comments on drafts of this report. We would also like to also mention and thank Craig Langenfeld from RALLY for the numerous discussions on Agile in Regulated Environments. We remain hopeful that future releases will more easily allow exporting artifacts required for product submission to regulatory bodies. Recently Dean Leffingwell has started a blog (Leffingwell, 2010/11) that offers information relevant to the challenges we had to overcome and we strongly recommend reading it, especially for people operating in regulated environments.

References