Group 4

Agile 2.0

Cory Ferro, Colby Kerwin, Mohammed Ibrahim

Q&A

**Group 1: What are some difficulties that growing web companies may face using agile methodology?**

One of the main disadvantages to a growing web company is shaking the consistencies and traditions that got them there in the first place. The instinct is to stick to what made you successful, and some companies kind of set aside the important fact that as your company grows, your infrastructure must be forced to adapt. Growing web companies meet bigger demands and more complex challenges, and so when the training wheels come off, the practices that got them rolling in the first place will not be enough.   
  
Furthermore, there is no discrete timeline of growth for a company, meaning the managing members don't get to sit down and say "today we grew from small to medium" or "large to x-large", it's a continuous and subtle growth, and the members of the company will have the biggest challenge detecting change over time - think of how a person only notices his or her own growth if they look at pictures. This corporate myopia creates an issue for companies looking to adapt. The best thing to do is to take as many "snapshots" as possible, not just financial statements, but also efficiency reports, developer velocities, customer satisfaction, etc. This will show where corporate growth is bottlenecking, and which parts of the corporate philosophy need to be tweaked.

**Group 2: Can agile methodology be used other than in the development of projects and software testing? If so, which projects would you say it would be best used for?**  
  
Although the concept of agile methodologies has a software/technology origin, I definitely think that there are components that can be and are being used in other industries. I have been to a few agile conferences where representatives from the non-technical side of companies like Carnival Cruise and Disney attended and explained how they have incorporated agile methodologies into their operational processes. Many of agile's communication constructs and its user-oriented, incremental focus have been adopted across several industries.

**Group 5: What if any laws or social decency are in place in regards to reuse of code? (Do social norms support anyone's reuse of the code due to their collective ownership, or is it frowned upon?)**  
  
I'm not sure if this question was meant for our group since we presented on agile methodology, but I will answer your question.   
  
The GNU General Public License, version 2 (GPL-2.0) serves as kind of a spectrum of permissions as they relate to open source software projects, starting with the most lenient, and getting more stringent as you go down the list. The author of the software may choose one of these licenses, and anyone who interacts with that software is prompted to agree (usually via confirmation of terms and conditions prior to download) and the user is thus subject to the agreement.   
  
The GNU GPL is really more of a constitution than a contract, because software is so different from project to project, that the burden of legality really falls on the proprietor. Arguing whether or not a GPL is violated, or whether or not the user was had sufficient exposure to the guidelines, is not a battle worth fighting. That's why big software firms have in-house legal teams dedicated to writing specific T&Cs and privacy policies for all their software, and then making sure that the legal docs are updated as the software is updated.

**Group 6: What would be the advantages of having executives that understand how programming works?**

I think it is more important for executives to understand how their product works rather than understanding how programming works. Many companies have a tiered development approach where there is a clear separation of planning and implementation. As executives are rarely involved in implementation, it is more important for them to be able to articulate clear objectives rather than worrying about the nuts and bolts of what goes on behind the scenes. All of upper management should understand the industry that they are in and the current trends but they do not need to be as worried about getting down in the weeds. One of the benefits of upper management understanding programming is that communication is easier. Their expectations will be more realistic and it will be easier to make decisions. However, there is value in having management personnel who can be focused more on the user rather than technical implementation.

**Group 7: While overcoming with the challenges of implementing agile methods. In what environment most likely will these agile methodology be successful?**  
  
Agile is not precisely a tangible concept - there are rules and restrictions, but the primary goals are fluidity, cross-functionality, and quick-thinking. My point is that agile is a concept that can be tweaked to work for a 5-member development firm or a company like Microsoft, which uses agile regularly. With that being said, I can argue that the best situation for implementing agile is a small- to medium-sized company with a workload growing faster than its personnel. In this situation, the key is to maximize on the efficiency of the developer and the team as a whole.   
  
Agile will help a team like this get the most out of each developer by gauging his or her productivity, making sure he or she is in the right position, and constantly monitoring progress of tasks. The efficiency of the team as a whole will not necessarily be an aggregation of the individuals' efficiencies, but more on whether or not the organizational structure serves as a good "petri dish" for the dev team. On the other hand, the individual's performance will depend on his or her ability to function well in that structure.

**Group 9: Agile development has a notorious disadvantage when it comes to the physical and psychological demands of the developers. Could increasing the amount of developers help this issue? Considering that more developers working for a dynamic project can in turn create more problems in other areas such as, for example, communication. What other methods can be used to prevent developers from suffering from such intense pressure?**

I began writing this response at the office at 7:15am, 7 hours after I left last night. Trust me, I hear you in regards to the psychological vice grip that is an agile company. However, there are several things for a scrum master to look at first when he or she notices a developer getting close to burning out.   
  
First of all, (me as the scrum master) am I underestimating hours for that dev's tasks? I'd look and see how many hours over the estimate that dev is working, and if the situation is chronic, I need to figure out if it's my fault for assuming tasks are smaller than they actually are, or that dev is just in the wrong place.   
  
But let's assume the dev is on the money when it comes to estimation.   
  
Am I simply assigning too much work? As a scrum master, I need to know what the 100% efficiency level is for that dev, and I need to monitor that. Let's say for this dev, the max he or she can handle is 35 task hours a week. I (actual me) tell my scrum master to start developers at 70% of their max efficiency, and increase it by 5% a week if they hit their mark, capped at 90%. I never want my devs to be forced to perfection, machines can't even do that. If one week, a dev drops below the margin of error, I ask my scrum master to give that dev 5% less, with a floor of 70%. If the dev drops below 70%, then I need the scrum master to find out what's wrong, because it's obviously not the work.   
  
This is my take on scrum, and the way I like to implement it at my company. It may look like punishment, but it's exactly the opposite - I want to account for the human element, sometimes devs need a break, sometimes they're ready for turbo mode, most of the time they're in the middle. Stressed out devs is almost always a result improper management.   
  
There are techniques during work with which to experiment to relieve stress. Pomodoro is one technique that my devs love, pomodorotechnique.com can explain it. Personally, I keep soft, dim lights in the office and (I know its fem) a couple candles, and play some trancy zen stuff as background music. Every day I feel more and more like the boss from Grandma's Boy (I even have my weird tea) but it's better than coding in a straightjacket.

**Group 10: Analyze LaunchCode, St. Louis, as a social phenomenon that uses Agile 2.0.**

This is a very unique idea. I worked for a staffing company for over two years and this concept never passed my mind. I feel like the technology industry has not embraced the idea of apprenticeships in the same way that other industries have. Although many technology companies have internship programs, there is usually not a huge emphasis on mentoring a coder who lacks experience so that they can get up to speed. I see that the major agile component that LaunchCode uses is pair programming. Based on most of the people that I have spoken to, pair programming is not widely accepted in the industry. Some companies have really jump on board with that methodology but it is definitely difficult to integrate that into traditional processes. However, I think that pair programming is perfect to accomplish the objective of mentoring new coders. I think that as a social phenomenon, this idea could really catch on. It may be worthwhile for companies to incorporate pair programming into their onboarding and training processes for new employees. This would not only benefit newer coders. It could benefit all new employees who need to get up to speed in regards to learning how the new product works and understanding the tools and processes that the development team uses. Overall, I think that this is could be a revolutionary concept that will be effective in the training process of new employees. I doubt that it will have long term effects that the individual companies other than helping new employees to become productive quicker.