Agile: Adopting a New Methodology at Harvard Business School

Susan Borges
Supervisor, HBS Applications Services Group - Harvard Business School
sborges@hbs.edu
Jennifer Gilmore
Database Business Analyst - Harvard Business School
jgilmore@hbs.edu
Sarah Edrie Oliveira
Director, Application Development Support Services - Harvard Business School
saoliveira@hbs.edu

Abstract

As academic institutions become more reliant on Information Technology for everyday work and learning, Harvard Business School needed to find a better way to meet the ever-increasing demands of the faculty, staff, and student body without increasing resources.

Adopting the Agile methodology was our way of responding proactively to these changes and in little over a year we have successfully implemented many projects with impressive results.

This paper will discuss how we were able to become Agile, our challenges, lessons learned, and what we gained from this change.

1. Project Background

Imagine turning an entire organization of over 100 IT professionals trained to measure twice and cut once, on its ear. Since its inception in 1996, the Information Technology Group at the Harvard Business School (HBS) used the Waterfall method of software project management. With a continually growing portfolio of projects and a fixed capacity, team members were pulled into up to ten different projects at a time, making it difficult to keep up with the demands of new project work while maintaining 170 mostly custom built applications in production. Since team compositions were continually morphing depending on who had available time, it was difficult to transfer knowledge from one project to another. Moreover, some thought was given to commonality of code but there was little motivation to share; therefore, reuse of valuable code was negligible at best. This very busy schedule, along with a huge demand for more products, made pleasing our partners1 very difficult. Best summed up by a fellow team member, our partners just expected our projects to be late.

2. HBS at a Glance

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td># of Employees in the IT department</td>
<td>100</td>
</tr>
<tr>
<td># of Faculty, Staff</td>
<td>1245</td>
</tr>
<tr>
<td># of MBA, Doctoral Students</td>
<td>1892</td>
</tr>
<tr>
<td># of Executive Education Participants</td>
<td>8133</td>
</tr>
<tr>
<td># of Agile Teams</td>
<td>4</td>
</tr>
<tr>
<td># of Ongoing Projects</td>
<td>15</td>
</tr>
<tr>
<td># of Current Applications</td>
<td>170</td>
</tr>
<tr>
<td># of Approved Projects</td>
<td>120</td>
</tr>
</tbody>
</table>

3. Implementing the Agile Process

After extensive research of various contemporary management systems, the HBS IT Group fully committed itself to the Agile methodology in early 2006. With a stronger focus on teamwork, communication and partner involvement, this meant a big change in the way projects and people were managed.

One of the big concerns for our implementation of Agile at Harvard Business School was how much input

---

1 At HBS we refer to our client base as Partners rather than Customers or Clients because of the collaborative working environment it fosters.
we could get from our partners. Some of our partners are very busy faculty members and high-level administrators. Would they be able to give a time commitment to their projects, so that stories could be ranked together and iterations agreed upon?

Another concern was how we would handle our large backlog of projects. Before implementing Agile, we usually had 100-120 projects in our queue. We often started many projects at the same time, believing that a little progress on each one was better than making so many of our partners wait for development time. It was common for some team members to be assigned up to ten projects at one time.

A few members of the team discussed these issues with some HBS faculty and the group agreed that the Agile methodology could actually save us from some of our issues. The Agile Manifesto was appealing – if we spent more time collaborating with partners, perhaps we would spend less time reworking our products in the last phases. We were hopeful that having partners more in touch with the day-to-day process of software development would help them to prioritize features and be more realistic about what could be achieved in a given amount of time. We sent two team members to an Agile conference to hear more and to find out if it might indeed work at HBS. We then had some consultants come to the campus for a week to discuss how an implementation might work at HBS. Could we take pieces of Agile, or should we implement it wholesale?

After many discussions and careful thought, we felt that a customized version of the Agile methodology could work, especially with our administrative partners. The HBS management team set out to educate and sell the idea to staff and partners through training, literature, coaching and trying a few practices. We held many meetings with our partners, to educate them on the benefits of Agile and to show them how much of an engagement it would require. The most exciting benefit to them was being able to see an application come to life piece by piece and to be able to choose which features were added as the project moved along. A joint decision was made to have one pilot development project serve as a proof of concept.

A few of the practices were tried in various projects and then a pilot team, known as the Red Team, was formed to work on a project using the new methodology. The project took six weeks to complete and the process used was continually revised to work best in the HBS environment. The partners were very engaged in the project, often attending the daily stand-up meetings and actively ranking the stories to be completed in each iteration. The project was deemed a success, due to the partner interaction, on-time completion, and satisfaction of the project team. The quality was excellent. Our partners got exactly what they expected, which resulted in far fewer late project changes and post-project fixes.

In the retrospective that followed the delivery of the project, the Red Team identified the following practices key to the project’s success:

- reducing the number of projects that a team worked on at one time, so that they could focus and avoid switching costs
- working together as a cross-functional team
- holding brief, daily stand-up meetings that involved the project team and partners
- using pair programming allowed the developers to leverage each other’s expertise

These lessons, as well as further coaching, helped fine tune the use of the Agile methodology at HBS. Three other teams were formed to encompass the rest of the development staff. Agile coaches came in for one to two week stays, helping each team get started and to offer advice for how best to implement Agile at HBS. After each team’s first project, the entire development team came together for a "coffee talk," where the best practices for each project were discussed. The teams talked about how estimation was done and how velocity was calculated on each team, what they found most and least productive about Agile, and how committed project partners had been.

Less then one year from the beginning of the implementation of the Agile methodology, most development projects are Agile. Each project gets a specific time box, and the number of iterations is decided. Our iterations range from one to two weeks. To establish velocity, the total number of hours available and estimations of how long each feature will take are calculated. We assume that seven hours of every forty will be non-project time, such as project reporting, meetings, and administrative overhead. Stories are then broken down into units, which are relative units of measure defined by each project team. One story may take three units, and another only one. Velocity is then calculated by how many units can be completed per week.

4. Increasing Focus

From our team discussions, we have found that it is better to have team members focused on the project at
hand. Today, each team member averages two projects, along with some maintenance activity.

Ideally each Agile team is comprised of one or more partners, two project managers, two fully-dedicated developers, two database resources, a quality assurance resource, and an applications advisor. From this team configuration, resources are used as per needed for each project (i.e. some projects require more intensive database work than others). The team stays together during the span of a project. We have learned that in some cases semi-permanent teams allow for more knowledge to be shared during pair programming sessions and stand-up meetings, although we may cross-pollinate the Agile teams to leverage knowledge across the group in the future.

Our partners are much happier knowing that when a project starts, it gets our full attention. It is completed much more quickly and the partner sees the progress. However, it is still difficult to keep our partners waiting to begin a new project, and Agile has not lessened our queue of 100-120 waiting projects.

5. Challenges

The challenges we have encountered can be broken down into three categories: physical space, team resources, and refactoring applications. Physical space proved to be the biggest challenge.

5.1 Physical Space

Agile methodology advocates teams co-locating, with the idea being that communication, and therefore productivity, increases when team members are in close contact. The two buildings we occupy are a converted dormitory and an off-site building that is within walking distance. How the teams are housed is not conducive to Agile work teams. We’ve moved team members into functional groups with each team having an open team space in which to collaborate. Each of the four teams is co-located, except for the Database resources. The Database Group still sits together in several offices, with two to an office. They are not in an open environment and need to move to be with a specific project team.

Of the four Agile teams, one is located off-site and uses some shared on-site resources. This makes it much more difficult to share Agile practices with the one isolated team, their adoption of Agile has been slower than with the other three teams.

Going forward, we have committed ourselves to finding creative ways to resolve the issues around physical space. There are plans to renovate one of our spaces so that it is more open and more conducive to pairing and collaboration.

5.2 Team Resources

Along with the physical resources challenges we have had to creatively manage, we’ve also had to deal with challenges on our teams. Specifically, resource constraints, role confusion, and work-life balance.

5.2.1 Resource Constraints

Database and Quality Assurance resources are partly dedicated to the team with many other projects on the side. There are not enough Database and Quality Assurance people to be allocated full time to each team. This creates a bottleneck for the team if a particular resource is not available. Each Database person also has a specific weekly allocation for maintenance and product support. At times he or she may have to ‘fight fires’ (high priority support where, for example, a high usage application becomes unavailable). While regular maintenance and support can be allocated and is expected within project teams, the high priority support issue requiring immediate attention pushes back scheduled Agile project activities.

We also have projects that "fall from the sky" and bump projects that are already in progress. A faculty initiative may mean quick turnaround for a high-priority project. For example, if a faculty member requires a new question be added to their course evaluations, it must happen by the end of term. Faculty projects often bump administrative projects. In addition, University priorities may quickly become our priorities if they are related to student privacy issues or the way that our data is shared with central administration. However, Agile, by dictionary definition, is the ability to respond quickly to change. The good news is that some of these projects fall under the purview of our partners and then they can re-prioritize when one of their projects needs to bump another.

5.2.2 Role Confusion

While it is very clear to the Developers and Quality Assurance what role they play in the Agile process, the Database group still has questions about the process that have gone unanswered. With all the support, maintenance, and non-Agile project work that the database needs to do, they are often caught firefighting or between Waterfall and Agile methodologies.
5.2.3 Work-Life Balance and Estimations
One of the other challenges that the teams face is inaccurate estimations. Though this is not completely new with Agile, a collaborative environment brings this issue to a head. Task estimations are something that has been off the mark—mostly under estimating. Instead of correcting the time needed for a task, the person would often feel compelled to work at night to complete it therefore creating a false representation of the time that was estimated. At times it seemed it was more important to keep moving forward rather at whatever cost which caused an imbalance in the work-life balance that Agile promotes.

5.3 Refactoring Applications
Having worked for so many years in a Waterfall environment has left us with quite a large number of applications that need to be refactored, application wise and database wise. The backlog of refactoring that needs to be done will be a big challenge that will need to be looked into further. To mitigate the risk created we are putting in the portfolio technical debt reduction projects and technical stories.

5.4 Concept Phase and Iteration Zero
In a recent Agile Coffee Talk, we discussed the difficulties in obtaining accurate estimations with only one to two hours of time allocated to understanding the project in question. We knew that our accuracy in estimating the duration of a project was suffering and, as a result, our projects were not completing on time. If we didn't have enough time to think about the project in detail, we could not ascertain what we did not know. We have decided to implement a time-boxed “concept phase” for our projects, where we allow the team to do some mock-ups to find where the difficulties might be. We also want to focus on architecture during this phase, so that we can create more sustainable applications. We have also implemented an “Iteration Zero” in which some of the technical stories are addressed. With these improvements, we expect that our estimation accuracy will increase dramatically, our maintenance tasks will decrease, as will our technical risks.

6. Project Retrospective

6.1 Happy Teams
Agile collaboration creates an environment that is pleasant and supportive. One team member wrote on our project Wiki, "[We] had to create a table for [the project] last week. Instead of doing it by communicating back and forth through emails and bits of conversation, we sat down in front of my laptop and mapped it out, wrote the script to create the table, keys, grants, and stuck it in CVS—it was a much faster way of doing it!! No going back and forth! The collaboration was great and we finished it in a fraction of the time we usually would!"

The work itself is more engaging, challenging and satisfying when developers work in pairs. The two developers get to solve problems together, and share their differing areas of expertise. Although developers are known to be solitary people, several of the pairs have nothing but praise for this style of working. Junior developers learn a great deal from senior developers and the senior developers enjoy the fresh problem-solving perspective from less experienced developers. The pairs are also not always the same, so that the tacit knowledge can become more widespread. One developer said, "Devs like working more closely with each other/teammates to collaborate on solutions instead of working in isolation."

6.2 Efficient Teams
Because of the iterative approach, we are able to produce more features within less time, which results in lower costs for HBS. The costs from project to project when our team members were working on as many as ten projects at a time, spending perhaps two hours on one project at a time, were astronomical. With a more focused development team, they can focus and complete an amazing amount of stories in a short time.

In addition, the projects actually end. We agree with our partners that a project will be time boxed and features that fit within that time will be implemented while other lower priority ones will not. Before Agile, the project would have a large concept phase, and a project plan, but in the end we may have "forgotten" important features, which would be added even in late stages. The end of a project would be pushed out again and again. As a result, each project took as much as twice the estimated duration, all while the partners were frustrated that we hadn't coded what they wanted. Now we agree to an amount of time and the features that make it in are highest priority and highest value to the partner. They understand the concept and the fact that they may need to save stories for a later development project if they don't rank high enough to be prioritized. This reduction in project time ends up
saving real dollars, as well as adds actual value to the portfolio.

Our increased coordination also means more sharing of quality code, which leads to even lower costs as developers can reuse code (such as a pop-up calendar for date choice, which has been re-used in a number of applications).

One team member described one quick solution, "We all went into an office to work out solution options on the whiteboard and to come up with a two-way partner conversation where we all contributed to a decision." Many skilled people working thoughtfully create a higher quality decision.

6.3 Creative Teams

The atmosphere is one of mutual trust and respect; the competence of all team members is high; and the work challenges are engaging. Because of our increased collaboration, more input from our partners, and shorter iterations, the environment lends itself to creativity and innovation. Two of the Agile Manifesto’s tenets, "individuals and interactions over processes and tools," and "working software over comprehensive documentation" free up significant time for creativity. Our processes help speed up the delivery of quality software, instead of bog it down.

The enthusiasm to make Agile work even better is still very much alive! In our last visit from an Agile implementation consultant, he mentioned how impressed he was with how far HBS had come in so short a time and how dedicated we are to moving even further with Agile.

6.4 Agile Coffee Talks

To help encourage cross-team collaboration, we hold periodic ‘Agile Coffee Talks,’ where all of the Agile teams come together to talk about their successes, challenges, and recommendations. We have mentioned some of the results from past Coffee Talks in the preceding paragraphs. In one of the last Coffee Talks, the group decided to set aside community-wide concentrated working times in response to recommendations made by the teams.

6.5 Satisfied Partners

Perhaps one of the best effects of the Agile methodology has been the increased communication with the end users of each of our products. Before Agile, the HBS development team had a reputation for working very hard but rarely delivering products on time. Our partners were generally satisfied with the work that was produced, but they felt that not enough work was done on their projects and felt that the significant delays detracted from the value of the work. Now, each feature request is ranked in a meeting with the partner. Then, there is a discussion of how many pieces (stories) can fit into the next iteration of the software and a line is drawn.

The Agile methodology has allowed us to respond to and even welcome changes in specifications at any point during the development process. This allows us to keep innovating to further the pedagogical needs of our faculty and the administrative needs of a very demanding student body.

A measurement of our success in adopting Agile was evident when our partners came up with a last minute change, wrote a story, and started reprioritizing other stories themselves without prompting from anyone in IT! The important feature may have been forgotten in the old Waterfall method, until the final release of the product. This would have meant costly rework and a delay of the next project. Instead, the change worked into the schedule almost seamlessly.

A palpable shift in the relationship that we have with our partners has been felt by everyone involved on the project teams. By exposing our process and getting them involved, they now appreciate and understand the work behind a project. One partner, for example, said that she couldn’t understand before why projects would be delayed with technical issues. While she might not understand the full depth of some of our technical issues, she can now see what’s involved and the team’s commitment in creating a high quality product.

7. Recommendations

7.1 Resource Limitations

- Limit the number of projects that the group takes on so each team member can become more focused on particular projects and decrease their switching costs from project to project.
- Improve portfolio management by developing expectations and working agreements with stakeholders so as to reduce ‘sudden’ high-profile projects.
- Set aside dedicated working times (such as Tuesday and Thursday mornings and
Wednesday afternoons) to allow team members to do a flow of continual work uninterrupted by meetings.

- Eliminate unnecessary meetings by co-locating a team and allow team members the option of declining a formal meeting, without penalty, if they deem it unproductive.

7.2 Space Constraints

- Maximize the use of space by tearing down some walls in the building to allow more freedom and eliminate the dorm feel of the building.
- Increase the appeal of teams’ using their respective team spaces by making it more comfortable to work in (i.e. better chairs and desks, more whiteboards and supplies readily available).

7.3 Further Recommendations

- **Monitor Velocity**—Efforts to better calculate the velocity of a project are needed. For example, teams were allocated x amount of time per week for maintenance work, which was exceeded every week and caused miscalculations. It was found that some team members would pad their estimations so as not to over promise a deliverable and be penalized. They should be encouraged without penalty to give their ‘real’ estimates so as to fine-tune the process. And the duration of a project should be closely watched and time boxed.
- **Continued Feedback**—Agile Coffee Talk sessions should continue to be held. There has been very positive feedback from these talks and it is felt by team members that management is listening.
- **Training**—Further training should be given in order for each team to refine their application of this methodology and to break any bad habits that may have formed. For example, losing sight of the bigger picture for the smaller stories we work on during each iteration. Training also extends to the teams educating their stakeholders about Agile and how it is applied.

8. Conclusion

Our biggest advantage in moving to Agile has been the excitement it generated in the community. Everyone was willing to try it out and worked hard to make this “ear turning” cultural shift a success. Agile allows us not only to continue to measure and cut but also to think about what we are measuring and cutting and what value the result will have to our partners, our teams and our portfolio. To date, we have successfully completed a number of Agile projects and teams are still excited and eager to fine tune the process. We are optimistic that using the Agile methodology will allow us to continue to produce higher quality and more efficient and valuable products in less time, and would recommend this process to other development teams who want to improve their efficiency, enhance their creativity, and keep their stakeholders happy.

9. Acknowledgements

The authors would like to thank Stefanie Horvath, Peter Leis and Naresh Jain for their help and support. We would also like to express gratitude to our teams, without whom this paper would be about gardening.

10. References


