Octopus: Agile software development facing our imperfect world

Eric Groise & Nicolas Mangin
Octopus Micro Finance Suite - OCTO Technology
eric@octopusnetwork.org, nicolas@octopusnetwork.org

Abstract
Agile practices offer very successful improvements for building software. Despite these great enrichments, we are trying to reduce the unknown part in software projects: the so-called ‘human factor’. We are still in the pursuit of the magic potion - that will help us solve issues like initial requirements specifications, parallel evolutions and change management. For Octopus we had to change people’s practices remotely in poor countries like Tajikistan and Afghanistan. Here is the true story of an agile project that had to adapt its practices drastically to meet real life conditions.

Oxus Afghanistan & Octopus teams

1. Octopus’s birth
When we started Octopus in November 2005, agile practices were already spreading at OCTO Technology. We had just successfully achieved the first official XP coaching mission for a bank. The arrival of these unusual “XP Guys” in the company accelerated the diffusion of this great agile attitude.

At the same time, OCTO decided to participate in the creation of Oxus Development Network, ACTED1’s subsidiary specializing in Micro Finance projects in Central Asia. As their technical partner, OCTO Technology has undertaken the task of defining Oxus IT systems. The Octopus project was born.

In the middle of 2006, this expert partnership gave birth to the first release of Octopus Micro Finance Suite (OMFS). To make sure that most people can benefit from our work, we have chosen the Open Source (LGPL) license. Then the real story began: using agile practices to manage a piece of software used in three distant countries by loan officers without any technical skills.

2. The Micro Finance ‘bank touch’
As usual, the first step of our project was to understand the functional environment. Micro Finance is mostly based on simple banking and community concepts that most people already know or can easily understand. And this sounds great! Because banking activities are something we master quite well here at Octo. We have already successfully completed many projects for the major French banks. So let’s do this project the way we’re used to doing it: healthy classic practices with a ‘bit’ of agility. But in fact just having a build factory and Unit Testing will not be enough.

It’s amazing to see how the real world complicates simple ideas. When you borrow money from a bank, things are quite simple: you know who will pay, how much money will be involved and when exactly you have to pay it back. But when you loan money in a small village in the middle of a poor country things are more complex. First you must manage “solidarity groups”, where each one’s behavior has consequences on all others. Also the borrower can pay later or sooner because walking three days from his village can involve

1 Agency for Technical Cooperation and Development
overcoming obstacles. He can also pay less than expected or sometimes more depending on what happened to him on his way to the bank. When you gather all these small user stories, your initial Microfinance vision becomes something much more complex. The fact is that lending 100$ in developing countries is much more complicated than lending 1M$ in our developed countries, a true paradox.

So we realized very quickly that if we want to master all this unexpected complexity, just a piece of agility would not be enough at all. We must go much further than just having the technical team using some XP practices (Unit Testing, refactoring and Build Factory). This time we must take it to the next level: Building Octopus in a fully iterative approach by including the business team in the agile process. So the first great decision we made was to switch to a truly agile project.

Nice decision! But what does it mean concretely? It means not only “using” agile tools for Octopus, but integrating these tools into an “agile process”. For example, before, our issue tracking tool (Jira) was a simple bug tracker (as very often in projects we cross). Now we fully based all development activities on it. Like this, Jira became the backbone of the project management, and the major means of communication between technical and business. Our natural tendency has been to take first what we easily get, rarely making the additional effort to get extra benefits from these agile practices. In this case, we used a simple tool and got extra benefit by integrating it with our agile process.

Another big change was the authors of these issues. Before, issues were mainly created by the technical team or the project leader. Today, the majority of these issues are created directly by the final users. These issues are now raising fundamental functional questions, resulting in rich discussions and clarifications between technical and functional people, whereas they were just enumerating the functionalities to implement before. This is now the actual heart & soul of Octopus: using an agile process to directly involve the final user in all project steps.

3. The power of Patterns

Due to Microcredit real world constraints, what looked initially as a simple piece of loaning software quickly became something much more ambitious. Microfinance software documentation is more often a book with hundreds of pages than few commented application screenshots, but we were determined not to follow the classic bank application specification path. We found a more agile way for capturing initial specifications. After many talks in Paris with the Oxus team experts, we realized how the “On site customer” practice is important. We started by building the initial software specifications quickly and efficiently. But after a few months our progress became slower and slower. Even if our local meetings were rich and productive, something was still missing in our mind. We concluded that we absolutely needed to oversee the project first hand in order to continue making progress. “On site customer” does not mean just being able to talk with your customer; it really means to meet physically the people in order to acquire the knowledge of the actual environment. You can’t make the good decisions if you are not able to imagine accurately the loan officer activity.

But a bigger difficulty was raised by this distance between us and our future users: communication. Sharing tons of Word documents is not efficient enough. Agile experts know this. But writing plenty of entries in an issue tracker system is not actually better. Validations take too much time, and having too much detail tends to hide essential ideas. We decided to use a practice we
already use at Octo, but for IT architectures: Patterns. One of the strengths of using these metaphors is to keep the focus on the key aspects of the problem. Instead of wasting lots of energy trying to agree precisely on each detail of a subject, we try to agree on generic functional designs that are then applied to several subjects.

The second power of patterns is its clarity. We try to use a simple vocabulary and graphical figures if needed in order to have the most understandable pattern description. Our goal is to be sure that everybody is talking about and understanding the same thing.

**Figure 1: Multi-level pattern**

So we took the plane in order to see the environment onsite and understand the day-to-day work of a Microcredit loan officer. (One of us spent one week meeting the people working at Oxus in Tajikistan). And instead of writing tons of documents or fighting around Jira issues, we focused on understanding the key patterns of this particular world. We name each one of these patterns and we then base all functional discussions around these patterns.

Without this pattern approach, we wouldn’t be able to avoid the big 500 pages specification book. It’s the supremacy of abstract design against concrete description. Of course, we had to land sooner or later on all the abundant real elements, but these patterns allowed us to delay this step, and keep focusing on the key aspects of the application functional design. Understanding is much more important than just describing.

4. Unit testing is great, but not enough

The solution we found for transforming these patterns into real/material specifications is inspired, and we are quite proud of it.

We all know that Unit Testing is a big idea, and a need for this testing is clear. The 1168 Unit Tests we run each time someone releases a new piece of code into Octopus’s repository are crucial. It’s a lot, but still insufficient because we are still missing some important areas. Today, we can’t imagine developing without using this very efficient “software insurance”. This technique is now used for all Octo projects.

But these tests still fail on one crucial point: providing direct accessibility to the functional expert. Of course he can view the test results reports which provide useful information about project progression and quality level. But he is just a spectator. To become a Unit Testing actor, he needs a key: master programming language. And for now, this key is not available to everyone.

On the other hand, there is an application that was originally considered a ‘nerd tool’ that completely succeeded in providing great results to non-technical people: Wiki. A Wiki is a website that allows visitors to add, remove and edit its content. Every time we used a Wiki tool on a project for building documentation or sharing work results it has been a great success. Everybody can participate as an actor, from the developer to the project manager.
Some clever guys had the great idea to mix Unit Testing tools and practices with the power of wiki. That’s why Fitness exists!

Developers still need to write some source code like in classic unit testing, but what makes the difference is that these tests are ‘configured’ by a Wiki, more specifically by simple tables that everybody can edit and understand. In these tables, the functional experts describe their business scenarios using real user data, and also provide the expected results. For example, they describe the loan’s characteristics, the expected loan schedule and the final expected accounts balance (see figure 2). What is great is that they can be expressed in many scenarios without requiring a developer, and they can see the result immediately by just clicking the “Test” button.

And check final balance, the principal has been moved from rescheduled loans (2032) to bad loans (2911). Note that 2971 liability account stores both interests and fees (44), whereas P&L accounts stores respectively interests (30 on 7028) and fees (14 on 7029).

<table>
<thead>
<tr>
<th>order</th>
<th>chartOfAccounts</th>
<th>account</th>
<th>type</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>2032</td>
<td>RESCHEDULED_LOANS</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>2911</td>
<td>BAD_LOANS</td>
<td>1000</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>2971</td>
<td>IPDPL</td>
<td>44</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>2991</td>
<td>LOAN_LOSS_RESERVE</td>
<td>144</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>6712</td>
<td>PROVISION_ON_BAD_LOANS</td>
<td>144</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>7027</td>
<td>PENALTIES_ON_PAST_DUE_LOANS</td>
<td>14</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>7028</td>
<td>INTERESTS_ON_BAD_LOANS</td>
<td>30</td>
</tr>
</tbody>
</table>

Figure 2: A Fitness test description

Of course, in order to make it fully helpful, each Microfinance feature contained in Octopus has to be demonstrated by story, a real life story described in the Fitness wiki.

We have been using this practice for more than one year, and the results are positive. The functional leader defines his functional requirements in the Fitness Wiki, and then the developer’s job is to make them work. Many critical debates between us and the Oxus officers have been solved by agreeing on a Fitness story. This tool is very helpful in ensuring that your application is working properly. We now have more than 65 business scenarios that make more than 12500 checks (assertions). Octopus’s fitness practice has now become the right path to follow for all projects at OCTO Technology.

5. The “Phantom software” fear factor

I personally love the moment when your application that you build over the course of several months using agile practices becomes something really alive: the famous Beta 1. It’s the magic pill that helps you get over the months of hard work. But unfortunately for the business users, this pill does not have the same taste. Being a developer is in fact an opportunity. When you are writing the software source code, you are directly using the raw materials to build the software. You can see it growing incrementally every day. But unfortunately your future user does not! Of course, as agile practices recommend, you tried to share the visibility of this progression with the stakeholders. But you were suffering from the lack of tangible results. There’s still work to be done in this area by building better business reports or business dashboards. So the view your future users have in mind is much more “Phantom software” than something real, something they can’t touch with their fingers.

In order to stop this increasing fear, we decided to build what we call a “ghost buster prototype.” What was unusual was that its goal was not to validate functional choices as usual prototype does. Its goal was just to show actually what Octopus was, and more accurately will be. We built this prototype while the real application had already been released several times! More than 10 iterations where already completed. Then we went onsite to physically show this Octopus prototype. It did
not helped directly to validate any functional choice, but it helped to significantly decrease the fear factor. We also returned onsite to show Octopus 1.0, the first release considered usable by the loan officers. We went there to make sure that everyone had the full vision of all Octopus could do. They understood where all this long time was spent. The ghost was definitely gone.

6. The unbreakable release manager

While we were finally solving this “phantom software” factor, another tricky issue was rising: the remoteness. Agile practices always recommend a very close relationship between the tech guys and the business guys. The best is to share the same offices, but also the Internet can help to reduce the distance drastically.

But the fact is that Microcredit specifically targets countries generally far from Octopus’s mother ship (Paris), and countries with very poor infrastructures. In Afghanistan for example, downloading a new release of Octopus (5 MB) means an hour of download. But downloading a new release of Microsoft SQL Express 2005 means hours or days. Also, the loan officers using Octopus are local employees, with very poor knowledge about information technologies.

So having a local “release manager” quickly became mandatory. His primary responsibilities are to validate the new release, distribute it to the local users, and centralize all users’ requests. But these responsibilities have many indirect duties. First of all they are the local Octopus contact, not like us sitting comfortably in our air conditioned offices, admiring the wonderful Eiffel tower through the window ☺. That means that when we release a dirty application (and it happens), he gets the sharp end of the stick, not us. Then he quickly forwards the amplified dissatisfaction to us. So believe me: this man needs to be rock-solid, or he will quickly refuse all your releases. You really have to find a “courageous” person that fully trusts your project. And luckily we did! Some XP Programming experts consider that bravery is the most important value for achieving successfully projects and we tend to agree based on our experience. If all the people you work with are not brave enough, their fear will quickly break the fragile trust relationship, and the project will break down.

Another quality required by a release manager is an open mind - he must get the real user’s feedback. When you ask one man to centralize these needs before forwarding them to you, instead of getting real user’s ideas, you will get your release manager’s vision. To avoid this problem you need someone who is able to understand user’s requirements, and who is able to accurately explain the requests even when they do not match his own ideas.

7. From Excel to Octopus: a very difficult move

We also faced the change management problem. Before our entrance, the loan officers were using ubiquitous software: Excel. The spreadsheet concept is a very powerful tool. It is one of these very simple ideas (just cells with numbers or formulas) that answer a very large variety of problems.

Of course, using Excel was becoming more and more unmanageable due to the increasing complexity of their activities, the increasing need for quality and the need for data consolidation. Octopus’s primary goal was to make their job easier, and they acknowledged it. But these spreadsheet where theirs, and delegating all this to some guys in Paris was not easy for them. Also, when a problem arose, they could solve it immediately in their spreadsheet, while they would now have to wait for the next Octopus release (every 2 or 3 weeks).

So the switch took time, a lot of time. Furthermore, whereas our goal was to reduce their work load, some of them decided to enter all the loans information twice (Octopus and the excel spreadsheet). It was quite a surprise when we learned it, and quite upsetting because we first interpreted it as a failure. But in fact that was a flawed analysis, because changing people is a really difficult task, and thinking they will switch in an instant was unrealistic.
In an agile project, transparency between all project actors is really imperative, and it really happened for Octopus. The required trust relationship between each actor is greatly founded on the visibility each one has on the other. So when we learned this “bad news”, it showed us that this transparency was working. With no trust between people, things can look superficially nice while real problems are creeping up. To make good decisions, it was important for us to have transparency to understand exactly what was happening there.

To enable the trust needed to get this transparency, we needed to focus on application quality. The most destructive trust killer is functional regression. While a user can accept that new software often brings new bugs, they do not accept new software that brings back a bug that was already fixed. Non regression is the minimal guaranty you must bring to your user. It means that writing Unit Test and Fitness stories for each bug your users find is a necessary effort. It’s a practice we are still working on today, because it’s really hard work when you are a small team. But we know we will have to fully apply this vital rule, sooner or later.

7. Conclusions

This great adventure taught us a lot and is still doing so. If we had to keep the five most important lessons, they would be:

- Very often, we hesitate to fully involve the business guys in the agile process. It’s so much easier to convince technical guys to apply XP practices! But making the effort to go further is really worth it.

- Reading many agile practice books and sharing experience with people is required before starting an agile project. But agile is also about innovation! Do not limit yourself to what you already know. And do not hesitate to adapt strongly agile practices to your needs or better: create new practices! Of course share it to the community. An agile guy must keep his mind wide open.

- Do not overestimate the power of agile process in delivering a first release of software. While you feel fully involved in the incremental process, your user is awaiting your application in the dark! It’s a very hard for him: don’t let him spend too much time there, or give him some light.

- Choose a multi-skilled release manager who is able to seek user ideas without passing judgment.

- Quality is vital, simply because lack of quality can quickly kill your project.

Acknowledgements

We wish to thank the OXUS team, and more particularly Pierre Baraud and Alexis Lebel who had to trust us for months without seeing any real application. They where our “Guinea pigs” who had to support our incomplete application releases for a while; and those who gave us so much help in making Octopus become the application we are so proud of today.

References

Octopus Web Site: http://www.octopusnetwork.org
Octopus Fitness Tests: http://octopusfit.octo.com
Octopus Jira: http://forge.octo.com/jira/browse/OMFS
Oxus: http://www.oxusnetwork.org
Octo Technology: http://www.octo.com