A Successful Application of the Agile Manifesto to Resource Constrained Projects with Open Specifications

Aneliya Gardjeliyska[1], Radu Dobrin[2] and Sasikumar Punnekkat[3]

[1] Sofia University, Department of Software Engineering, Sofia, Bulgaria
aneliya@abv.bg
[2, 3] Mälardalen University, Software Engineering Laboratory, Västerås, Sweden
{radu.dobrin, sasikumar.punnekkat}@mdh.se

Abstract. In this paper we describe how agile manifesto can be applied when a specific software solution is needed in a resource, i.e., time- and manpower- constrained environment, while the specifications are not detailed, and are prone to changes. The case study, to which this method was put on test, was an online application framework for the EU-funded EURECA project, which deals with academic mobility at all levels (e.g., students, researchers, and staff) between 16 different universities from countries in Asia and Europe. The organization of this project turned out to be a herculean task during the first call for applications. Subsequently, we developed an on-line application management framework with limited resources within a short span of time following the principles and practices of the agile manifesto. The stages and processes that were used during development are discussed in this paper, both in general and for the actual application in the EURECA framework development.

Keywords: agile manifesto, resource constraints, open specifications, international colaboration, mobility

1 Introduction

There are a lot of different projects in the software world – ranging from small to big, from well funded to not, from personal to corporate. We are going to focus our attention on one distinct type – projects that are extremely time-pressed and which have specifications prone to changes, and, on top of that, are developed by a very small group of people.

The need for such software systems usually arises in order to address the automation of a repetitive task that requires a lot of time and attention. Because of the recurring nature of the task, one of the goals is to build a working solution between one occurrence of the task and the other. This sets the first constraint of such projects - time. Another aspect is the specification of the needed software solution - there is a general idea of what needs to be automated, but the exact way of doing it, and on what scale, is not known. Finally, as the development of these projects needs to be organized in a very short time, the number of people involved in the project tends to be small.

For a more traditional software development method, these constraints may look a bit too much to cope with. However, if one uses a method tailored around the agile manifesto, the project has a good chance of succeeding.

Let's have a look at the agile manifesto which describes major principles of importance for such projects:

"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- *Individuals and interactions* over processes and tools
- *Working software* over comprehensive documentation
- *Customer collaboration* over contract negotiation
- *Responding to change* over following a plan

That is, while there is value in the items on the right, we value the items on the left more."[1]

The time in such projects is a critical component and it cannot be wasted on processes and tools. Hence, more emphasis must be put on communication between developer and manager/customers. For the same reason, the documentation has to be left in the background and the emphasis must be put on building a working system rather than following a plan.

Through our experience in creating the online application management framework for an important European project (EURECA)[2], a number of major stages and ongoing processes were identified. The workflow in the development process is illustrated in Figure 1.

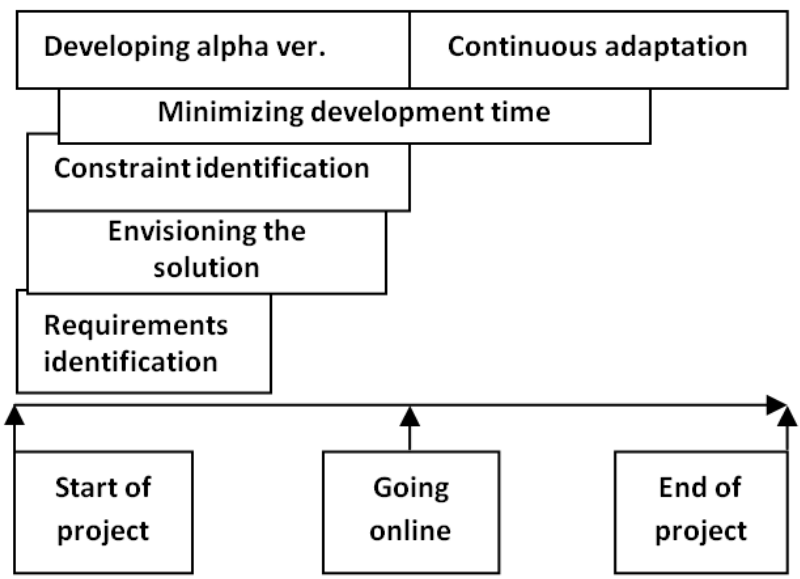

Fig. 1. Development process workflow.

We are going to discuss why these stages are important in general, when resource-limited small-to-medium projects with open specification need to be developed. We will also present the specific application of these methods within the EURECA project online application framework.

2 The EURECA Project - Overview

The EURECA (European Research and Educational Collaboration with Asia) project, funded by the European commission, aimed to establish a Eurasian academic mobility network, for achieving excellence in research and education in a global context. The consortium consisted of 16 prominent educational institutions (9 from Europe and 7 from Asia) and was coordinated by a European partner. With a funding of nearly 5 million Euros, EURECA project facilitated more than 300 students/researchers/faculty from Asia/EU to conduct a part of their study at one of the partner institutions. The primary goal of EURECA was to help top talents from Asia to achieve highest levels of excellence in research and education.

The EURECA project had some specific constraints - accepted thematic fields, maximum number of applicants coming from EU/Asia, time period of the mobility depending on the mobility type, eligibility, etc. Therefore, the coordination of the EURECA project was not an easy task. When the project started for the first time in 2008, most of the work was done manually leading to a low-efficient and time-consuming process. Eventually, the need for a software framework to facilitate this task was recognized. The EURECA online application framework had to be developed under two specific constraints - limited resources and not fully specified requirements, prone to changes. The limited resource constraint had two dimensions. First of all, the time was scarce, as work on the framework started in mid January 2009 and it (or at least part of it) had to be ready by March 2009. Secondly, the man power was also limited – only one student was developing the framework leading to the impossibility of parallelizing the work. The requirements of the framework were somewhat vague, as in the previous call almost everything was done manually, and, other than an online application submission, not much thought was given on how the process can be optimized from the perspective of all involved parties. With these constraints, it was clear that we would head for agile software development and management.

The key people, who started the software project, consist of one of the EURECA project coordinator (called “mentor”) and the developer (master thesis student). The rest of the people involved in the project were the global coordinators of the EURECA project, the universities’ coordinators (“partners”), and all people who applied to the mobility, i.e., the “applicants”.

It is also worth to mention that the mentor and the developer were working remotely, as the mentor was from the EURECA coordination university, whereas the student was from one of the partner universities. Nevertheless, this agile way of work was made possible through daily online communication between mentor and developer, as well as between the mentor and the other interested parties.

3 Identifying Key Requirements

For the success of any project, it is crucial to know the basics of it, i.e., what problem the project is aiming to solve. Very often, a solution to the problem already exists, but it is not (fully) digitalized and involves a lot of manual and tedious work. The mentor gathers all the information that is available on the

problem's field, and discusses it with the development team. They have to arrive to conclusions as to which parts of the processes can be automated, and which parts need the attention of a human being. To put things into context, we will describe how we applied this to the EURECA project. After analyzing the management results of the first EURECA call, the following key issues were identified:

- applicants must enter data in a consistent format
- partners should be able to publish mobility capabilities, i.e., their local availabilities for mobility within the project
- coordination should be distributed between coordinators and partners
- applicants should be able to accept/deny a distribution and, in case of acceptance, automatically obtain an award letter

The discussion of the first EURECA call also gave us insight on the different stages of such a call (Figure 2).

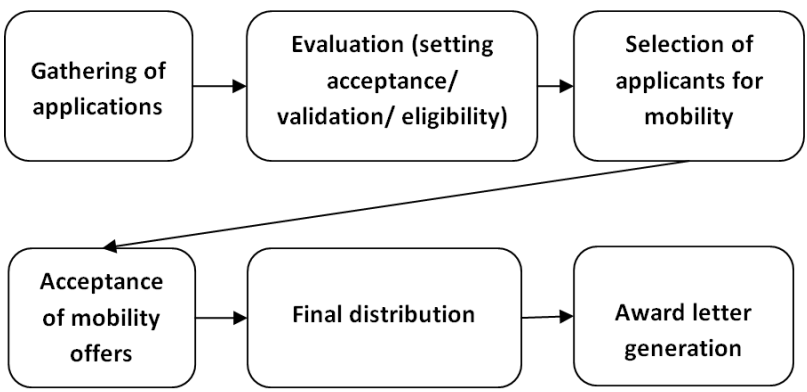

Fig. 2. Stages of one EURECA call.

4 Envisioning of the Software Solution

While the key requirements are being identified, it is possible for another process to be started – sketching the basic parts of the software solution. Important questions, which need to be answered together with the mentor, are: Who is going to use the system? What rights should the users have? What problems will be solved and how?

For the EURECA project, the involved parties were the applicants (both students and staff), the partners from the different universities, and the project coordinators. A key issue is that the three types of users must have completely different "rights" when working with the system. Consequently, we realized that we needed three different "modules" for the three types of users - applicants, partners, and coordinators. We used a database for storing all information and thus enabling the interaction between the different users.

5 Identifying and Overcoming Key Technical Constraints

When designing a brand new software system, it may seem that there are no technical constraints and that the development team may use whatever they want (if it is within the budget of the project, of course). However, usually some technical constraints do exist and it is important to identify them at the early stages of the project. These constraints typically affect the choice of programming language, the database server, etc.

For the EURECA project, one of the main issues we had to take into consideration was the capabilities of the coordinating partner servers on which the framework would be hosted along with other existing applications. The servers were running an Apache 1.0 web server and could host PHP v3 applications using MySQL v2 database. However, a major upgrade was planned for the servers within the next 2-3 months. Therefore, the EURECA framework had to be capable of running on both the current servers and the new ones, as it was crucial to ensure a smooth migration.

To address these issues, we decided on the following work pattern. The local implementation would be done on a local computer, running the latest version of WAMP, which in itself consists of the latest versions of MySQL, PHP and Apache. After a feature was ready and to be "tested", it would be uploaded on the coordinating partner's server in a location accessible only to a handful of people. Finally, we would test the new feature in the production environment, and fix any issues arising due to the use of older versions of the software products.

6 Minimizing the Development Time

Creating a software solution in a resources-constrained environment makes you value mostly one thing - the time for the development. However, how exactly to minimize this time, is a challenging task.

What we did for the EURECA project was to use open source solutions where this was possible. We had to be careful about the choices we made, because we didn't want to compromise on the security of the framework, or its design and usability. Another aspect that needed to be considered was how familiar we were with them and what learning curve they had.

After careful research, we decided on using the following free software solutions: MDB2 PHP module for working with the DB[3]; Smarty template library for separating the view from the controller in the MVC paradigm[4]; Yet Another Multicolumn Layout(YAML) for creation of the web pages [5]; SVN for keeping track of the development changes[6].

The first three products laid the fundamentals of the framework for the EURECA project and, through the development process, we continued to use other, smaller open-source or free packets [7][8]. Using these ready-made and tested products really boosted the speed of development and allowed us to focus on developing the new and unique functionalities of the framework rather than spending time on solving common tasks.

7 The Alpha Version and Going Online

Once the targets of the project are clear and one knows what the major software building blocks are, it is time for the real development to begin. It is important that the alpha version of the software stays focused on the main issues and provides the basic solutions to them. Of course, the "big picture" should be kept in mind, but trying to develop everything in the first version is just not possible.

For the EURECA project, we focused first on the applicant's module, because we had a firm deadline on when the gathering of the applications had to start. The partners' module was the next one to go. With the login, registration and edit account functionality already developed for the applicants, it was a simple task to modify the existing code to work for the needs of the partners, and then adding the specific functionality. Finally, with about two days left before going online, we built a small part of the coordinators' module.

For keeping track of all the requirements as well as bug fixes, we created a file similar to the SCRUM product backlog. However, unlike SCRUM, our file was a simple text file with explanations of the problems/features on each line of it. The information for the file was gathered every day during numerous Skype discussions. This file was kept in the SVN repository along with the rest of the code, and thus we were able to see at a glance what was done in each revision.

During the development of the alpha version, we had continuous uploads at the coordinating university's servers in order to check the framework's behavior at the actual production environment. This allowed us to have a pretty successful launch without any major problems.

8 Continuous Adaptation During Real-Time Usage

The release of the alpha version of a software framework is a major milestone in a project. Together with going online, it allows the developers and mentors to start receiving feedback on how useful the system is. Inevitably, the need for changing parts of the framework, as well as adding new functionalities in order to improve the user experience, is recognized. One of the big issues that arise during this process of continuous changes is the problem of updating the system during its usage.

During the EURECA application development, we used a new technique which we call "*pipelining*". Just like in hardware pipelines, we worked and delivered the application framework in parts, in a sequential way. The ability to divide the entire system functionality in chronologically ordered manner enabled us to improve the development efficiency. Such a "divide and conquer" approach allowed us to look at smaller chunks of code, with less clutter. In this context, by "*development efficiency*", we mean both reduction of development time, as well as improvement in the quality of the product. When a bug (or the need for a new feature) was identified, we did one of two things - either fixed it almost immediately or left it out for implementation in the next EURECA call. This allowed for really rapid growth of the framework. This approach of frequent releases, together with backing up our database, proved to be pretty safe.

By this pipeline technique, we managed to add a lot of new features to the coordinators' module - distributions/validations/eligibility, coordination tasks,

statistics, group e-mails. At the end of the EURECA call, we also enabled the applicants to accept/deny their distributions and ultimately receive their personalized award letter.

9 Post-development Experiences

Some of the most interesting aspects from the lifespan of an application are the various post-development experiences, which allow the whole team to assess how the application is used, what are the common pitfalls and what needs to be improved.

For the EURECA project the main source of feedback was provided by the "Contact us" forms. The one major bug - invalid registration of applicants due to time race condition - was identified quickly thanks to feedback from the applicants' module. Furthermore, we were able to tell that our fix of the problem worked - not only by monitoring the applicants' activities, but also by the fact that we did not receive any further complaints.

We also received a number of frustrated questions from the applicants - some have forgotten the login page URL, some have forgotten their email/pass and some could not figure out that the application was automatically submitted upon saving. Although, at first glance, these questions were somewhat unexpected to us, they provided an insight on the user experience and we were able to pinpoint the parts of the framework that needed rework. Therefore, we added better detailed messages to help the applicants, as well as a "Frequently Asked Questions" page on the website.

We also received several inquiries from partners, as well as few recommendations. This feedback helped us to improve the overall process of application ranking and setting of eligibility and to add more functionality for the partners.

10 Results and Statistics

At the end of the 2009/2010 EURECA call, we had a functional online framework that provided the most important tools for application gathering and distribution. The applicants were able to register within the system, to apply for mobility, to check their status, and, ultimately, to accept (or reject) the mobility that was proposed to them. At the same time, the system provided the needed checks for confirming the constraints for EURECA – three types of mobility, length of mobility depending on selected mobility type, and various other checks on the input values. The partners' module offered the functionality of registering new partner users and processing of outgoing and incoming applicants. The framework was keeping track of all partners' activities and thus the partners were more careful about their validations, eligibilities and acceptances. The coordinators' module provided different coordination and administration utilities – finishing the registration of new partner users, management and coordination of applicants, sending of mass e-mail messages, answering questions sent by the other two modules and showing statistical data.

Other important aspects of the newly developed framework are the various non-functional qualities that were implemented - configurability, security, availvability, deployability, and maintainability. For each module, the various configurations were kept in a dedicated configuration file. The security

was achieved by preventing SQL injections (using MDB2 binding capabilities), preventing XSS attacks (via Smarty "escape" filter) and locking users for a specific time after a specific number of invalid login attempts. The system was available almost 24/7 because the frequent releases meant less code for upload and thus less maintenance time. As the framework was developed for both older and newer versions of Apache/MySQL/PHP, its deployability requirements were not very rigorous. Finally, by following the agile principles, the code of the framework is easy to understand, while trace and adding of new features, or fixing small bugs is easily performed.

During the 2009/2010th EURECA call, approximately 550 applicants registered within the system. More than 320 of them filled out the application form, and 145 were selected for mobility. The partners had an easy way for validating, accepting and setting eligibility to the applicants. The coordinators were well aware of both applicants' and partners' activities throughout the call, and could react in time if required. Furthermore, the framework was used for a second time in the late 2009 when 1059 applicants registered within the system and 777 new full applications were submitted - marking the highest number of applicants in the history of the EURECA project.

Although the EURECA online application framework was successful and lessened the burden of coordination, a number of issues can be furthers addressed. First of all, the partners should be able to publish the capabilities of their universities. This will help the applicants to determine which university is most suitable for their mobility and thus improve the results achieved by the EURECA project.

Another possible improvement is the development of automatic distribution of the applicants. The online application framework gathers various data – applicants' mobility type and preferences, partners' verification and acceptance ranks, eligibility. Based on that data, together with the constraints from EU(total number of applicants, distribution within countries, etc.), it would be possible to provide a way for automatic distribution. This functionality will really make a difference and decrease the amount of work for coordinators.

11 Conclusions

Our own developed online application framework fulfilled most of its requirements and allowed a smooth process of application gathering, ranking and distribution.

One of the key elements for the successful development of the framework was the agile way of work for both development and management. The agile manifesto made us stay focused on our goals and the minimization of development together with pipelining allowed us to deliver a working framework in a short time. Because of the limited man-power, we were unable to parallelize the development work, but we did parallelize the processes that were ongoing during the development. This way, we did not require full knowledge about the requirements before we could start sketching the skeleton of the framework.

It is also worth noting that, while the created framework was specific to the EURECA project, it could be easily adapted to other mobility programs. This is possible because the key requirements of the student mobility problem were identified at the beginning, and they lay at the basis of the framework. This

serves to once again prove the point that a good understanding of the problem area is needed at the start of any project.

References

1. K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. C. M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas, "Manifesto for agile software development", http://agilemanifesto.org/, 2001
2. "Eureca project", http://www.mrtc.mdh.se/eureca, 2009
3. L. Alberton, D. Convissor, D. Coallier, A. Fazelzadeh, L. Smith, and P. Cooper, "Mdb2", http://pear.php.net/package/MDB2, 2004
4. M. Ohrt, A. Zmievski, M. Mohr, J. Boots, A. Holz, and F. Kromann, "Smarty: Templating engine", http://www.smarty.net/, 2002
5. D. Jesse, "Yet another multicolumn layout", http://www.yaml.de/en/home.html, 2005
6. B. Collins-Sussman, B. W. Fitzpatrick, and C. M. Pilato, "Version Control with Subversion", O'Reilly Media, June 2004
7. S. Langridge, "sorttable: Make all your tables sortable", http://www.kryogenix.org/code/browser/sorttable, 2007
8. H. Ortega-Hernández, "Datepickercontrol", http://dali.mty.itesm.mx/~hugo/js/datepickercontrol/