Using Agile Methods to Implement a Laboratory for Software Product Quality Evaluation

Javier Verdugo[1], Moisés Rodríguez[1], and Mario Piattini[1,2]

[1] Alarcos Quality Center, Paseo de la Universidad 4, 13071, Ciudad Real, Spain
{javier.verdugo,moises.rodriguez,
mario.piattini}@alarcosqualitycenter.com
[2] Institute of Information Technologies and Systems, University of Castilla–La Mancha,
Camino de Moledores s/n, 13051, Ciudad Real, Spain
Mario.Piattini@uclm.es

Abstract. In this paper we discuss how we at Alarcos Quality Center implemented AQCLab, the first laboratory in the world to be accredited as meeting ISO/IEC 17025 for software product quality evaluation based on the ISO/IEC 25000 series of standards. We implemented AQC Lab following agile principles by means of an adaptation of the Scrum methodology. This work method helped us to progress in a challenging context which had several similarities to software development, where the requirements were uncertain from the start.

Keywords: Software, quality evaluation, ISO/IEC 25000, SQuaRE, agile implementation, accredited laboratory, ISO/IEC 17025.

1 Introduction

Alarcos Quality Center (from now on referred to as AQC) is a Spanish company that was spun off from the Alarcos Research Group at the University of Castilla-La Mancha in 2008. It was founded with the goal of providing its customers (software factories and development departments, as well as software acquirers) with software quality assurance services. Though AQC is relatively young, we have over fifteen years of experience in software quality research that has already been carried out by the Alarcos Research Group.

After several projects that involved software process improvement, we realized that, though good development processes are of great help in the effort, they do not always lead to quality software; we became aware that the best way to evaluate quality in software products is by measuring and evaluating their own characteristics, not those of the processes followed.

That is why we decided to focus our work on developing a new service in an area that was not as well-known and widespread as others in the software industry: Indeed it is still not so widely-recognized, even now; we are talking about software product evaluation.

G. Cantone and M. Marchesi (Eds.): XP 2014, LNBIP 179, pp. 143–156, 2014.

By 2010, the new series of International Standards ISO/IEC 25000 [1] (known as Software product Quality Requirements and Evaluation - SQuaRE) was still in an early stage of development. We decided to take ISO/IEC 25000 as the basis for our software quality evaluations, even though the main standards in the series – the quality model and the evaluation process – were still under development. Four years later, SQuaRE is still being developed, though it has matured considerably and most of the main standards of the series have already been released. These include the quality model –presented in ISO/IEC 25010 [2] - and the evaluation process –defined in ISO/IEC 25040 [3].

At the beginning, there was a fair amount of uncertainty about how to deal with the implementation process, as it was a rather complex task in a not very well-known ground that involved a lot of research and experimentation. Right at that point, we knew we would have to:

- Develop a quality model. Starting from the quality model defined in ISO/IEC 25010, which specifies only top-level quality characteristics and their sub-characteristics, it would be essential for us to identify metrics and define how to aggregate their values to evaluate the top-level elements of the model.
- Implement an evaluation framework. We would have to identify tools that provide measurements for the metrics defined in the quality model. We would also need to develop a tool that takes those measurements and aggregates them according to our criteria so that we can obtain quality assessments for the top-level elements of the model.
- Define the evaluation process. Based on the evaluation process defined in ISO/IEC 25040, we would have to decide how to adapt that process to our circumstances.

The specific requirements to implement those three main work products were not totally clear from the start, given that it was difficult to define the scope of that endeavor completely. We thus realized that we would have to identify those requirements and deal with potential change along the way.

At that time, agile methods and techniques for software development had been around for a few years, and after a slow but steady rise and spread they were starting to become really popular in the industry. One of the most popular agile methods, Scrum [4], was a great exponent of the impact that the agile trend was having on the industry.

Seeing that agile principles addressed the same problems we had (dealing with uncertainty via evolutionary development, as well as providing a flexible response to change), though in a different context (software development), we decided to study which of those practices could be applicable and useful to our case. For this purpose, we took several training courses and workshops on Scrum that helped us to understand it better and get a better vision of the framework as a whole. Once we had a better knowledge of Scrum, we decided to adopt some of its practices and adapt them to our own objectives.

One month after we started to work on our software product quality evaluation service, and while researching other standards related to evaluation, we found out about ISO/IEC 17025 [5]. This standard specifies the general requirements for

laboratories to carry out tests and/or calibrations competently. To meet those requirements, laboratories have to implement a management system for their quality, administrative and technical operations.

Accreditation complying with ISO/IEC 17025 means the formal recognition by an accreditation body of the technical competency of the laboratory and its capability to provide correct and trustable results. In this regard, accreditation to ISO/IEC 17025 differs from certification to ISO 9001, which solely confirms that a company adheres to, and operates under, a documented quality system. To that end, accreditation bodies perform a thorough evaluation of laboratories, confirming that they:

- Count on qualified and experienced staff.
- Have the necessary equipment and infrastructure for suitable performance of their activities.
- Employ suitable and validated work methods and procedures.
- Perform techniques for quality evaluation of results.
- Inform their clients about test results in a suitable manner, providing clear and precise reports.
- Adhere to, and operate under, a quality system.

We considered that we would be making a valuable contribution in implementing a laboratory that would carry out tests consisting in the evaluation of software product quality; that is how AQC Lab emerged. We decided to pursue laboratory accreditation for several reasons:

- It would guarantee the integrity and competence of AQC Lab in its performance of software product quality evaluations.
- It would be a distinguishing feature and a key factor in keeping an edge over competition.
- Laboratory accreditation would result in an internationally-recognized service, as ISO/IEC 17025 is the best-known and most generally-accepted international standard for laboratory evaluation. In addition, accreditation bodies from different countries co-operate under multilateral agreements.

Implementing a laboratory that complied with ISO/IEC 17025 resulted in a whole new set of requirements, in addition to those we had already identified in relation to developing our software quality evaluation service. To meet the requirements of the laboratory accreditation scheme, we had to implement and document many different processes (both technical and administrative), produce formats, and keep records that documented and showed how those processes were carried out.

After a period of a year and a half of implementing, testing and validating our evaluation method, we carried out the first software quality evaluations for customers. Six months later, in 2012, AQC Lab became the first laboratory in the world to be given accreditation to perform software quality evaluation tests under ISO/IEC 17025.

The rest of the paper is organized as follows: in section 2 we discuss the two approaches to the adoption of Agile methodologies, which are either following them strictly or adapting them to fit the context of each project. In section 3 we describe

how we adapted Scrum to implement AQC Lab. Section 4 presents the conclusions of the paper, describing what we found most useful in our adaptation of Scrum.

2 Adapting Agile Methods

Since the emergence of Agile methods, there has been dispute among Agile advocates over the issue of whether to adopt methods "by the book", or rather to adapt them to serve the specific context of each company or development team. Even though nowadays this dispute has been overcome for the most part, there are still some practitioners that hold opposing views regarding this matter.

On the one hand, there are Agile advocates, commonly known as evangelists, who encourage all projects to follow every single practice of the Agile method in question to the letter. They argue that adopting the method as a whole is the only way to take full advantage of it, and any deviation from what is established by their authors would result in not realizing its full benefits. A quote from Kent Beck about XP practices [6] sums this reasoning up: "No single practice works well by itself; each needs the other practices to keep them in balance".

One of the fathers of Scrum, Ken Schwaber, coined a term for any deviation from the rules, roles and time boxes established in Scrum: "ScrumBut" (a term that gained popularity; some people later turned this into the more humorous "ScrumButt"). A ScrumBut can therefore be considered as an inappropriate variation of Scrum that hampers the team from getting the most out of it. Schwaber explains that ScrumButs follow the pattern (*ScrumBut*) (*reason* or *excuse*) (*workaround*). An example of this would be "(We use Scrum, but) (having a daily scrum every day is too much overhead,) (so we only have one per week.)". This example shows a kind of adaptation that negates the advantages of Scrum. In this case, the tailoring leads to not knowing the real progress of the sprint at the right time, in the right way; that in turn, leads to the possibility that the goals established for that sprint may not be met.

On many occasions, ScrumButs have their origins in a dysfunction in the development team and its inability to fix it. This results in the modification of the method, not because that is what is intended, but because the bad habits in the team do not let them find the way to adopt the method correctly.

On the other hand, an increasing number of practitioners and researchers, as stated in [7], argue that Agile development methods and practices should be adapted to fit the context in which they are adopted. These authors contend that, as with any other kind of adaptation, a tailored method may indeed not represent a reasonable adaptation of the original method. The "wrongdoing", however, is not in the act of adaptation itself, but rather in the nature or scope of the adaptation when it is not done suitably. They consider that being restrictive in adapting Agile methods is a kind of a paradox, because, as Conboy and Fitzgerald conclude in [8], "the very name "agile" suggests that the method should be easily adjusted to suit its environment".

This approach to Agility is based on the idea that a project cannot be viewed as an independent part of its surrounding context. Rather, the method followed to manage the project is affected by the interaction of the development team and their

organizational culture. It is difficult to keep any element that is external to the method from not affecting it in one way or another. Because of this, the practitioners that follow this approach stand for understanding how Agility can be adapted in context and take advantage of that situation. In this case, the main focus is on adapting the methods in a way that makes sense and improves the performance of the development team. This can only be done by understanding really well the purpose of the practices that will be adapted, introducing only the changes needed to make them work better in the context of each specific project or development team. Otherwise these adaptations, in the case of Scrum, would become negative Scrumbuts; the kind that make a team's performance worse.

There are practitioners and researchers, and we count ourselves among them, who believe that some of the Agile methodologies can even be adapted to other environments outside software development. A good example of this would be Scrum. This framework can be, and actually has been, adapted to different contexts other than software development, due to its strong focus on project management and its independence of specific technical practices.

For example, in [9], the authors present Score, an adaptation of Scrum to manage the mentoring of students in the context of an academic research group. The authors claim that ever since they have been carrying out some of the practices of Scrum, especially the daily scrum, the mentoring has been more efficient, and both the mentors and the students have benefitted from this new approach. For mentors, it is now easier to keep up-to-date with their students' progress, and when students are struggling, it takes less time to address what is not going right. Authors assert that students say they are more productive, more enthusiastic about research, and have better interactions with other students and with their adviser, feeling there is a real sense of community in the group since they began to use Score.

In [10], the author discusses how they applied an agile methodology in an academic environment, and provides insights for non-software industries on how agile is not a set of rigid rules, but a philosophy that can be applied to get maximally effective results with a mindset for continued change.

The authors of [11], among whom is Jeff Sutherland –co-creator of Scrum, together with Ken Schwaber -, describe how Scrum has been adopted in the sales and account management teams at the company iSense in the effort to take more control over the sales process they carry out. They conclude that implementing Scrum has led to escalating revenue and a sustainable competitive advantage.

On reading [12], we see how the author describes the experiences with Agile methods in a marketing department, as well as the series of adjustments they had to make to overcome some problems they had during the first months of adoption.

On a more exotic note, [13] describes how an Italian company producing luxury bathtubs and showers adopted Agile and Lean methods in many departments of the company, explaining how they adapted them to a non-software context.

The growing importance of Agile methods in project management is also reflected by the fact that the Project Management Institute (PMI) has developed a certification for project management practitioners who are adopting Agile approaches in their projects. This certification, known as PMI Agile Certified Practitioner (PMI-ACP),

recognizes an individual's expertise in using agile practices in their projects, while demonstrating their increased professional versatility through agile tools and techniques.

In the next sub-section we set out how we adapted Scrum in the implementation of our laboratory for software product quality evaluation tests; this is an endeavor that not only involved software development, but also process implementation, as well as a great deal of research.

3 Implementing AQC Lab

Although Scrum was conceived as a software development framework, it centers on management practices. Being involved in the software industry, though not developing software ourselves, we at AQC saw that Scrum could be applied in contexts other than software development. In our case, we saw Scrum would be suited to our purpose of putting a software quality evaluation service into operation, which would later expand and turn into implementation of a laboratory, AQC Lab, accredited as complying with ISO/IEC 17025 for conducting software quality evaluation tests.

Implementing AQC Lab involved different high-level tasks that would in turn encompass more specific tasks:

- Defining a quality model based on ISO/IEC 25010. As the standard only defines the high-level elements of the model (quality characteristics and its sub-characteristics), we would have to define which metrics affect the characteristics and sub-characteristics. We would also need to specify how to aggregate and combine their values so as to obtain a reasonable indicator of the quality of the software evaluated. For this purpose, we would define a hierarchical model and the methods or functions for obtaining values of higher-level element from the values of the elements on lower levels. Initially, we centered on the characteristic of Maintainability and its five subcharacteristics.
- Defining an evaluation process based on ISO/IEC 25040. We would have to define the steps to take, along with the specific way to carry out the activities of the evaluation process described in ISO/IEC 25040.
- Developing an automated evaluation framework. Once we had decided which metrics would be part of the quality model, we would need to look for tools that allowed us to get their values from the products analyzed. It would also be necessary to develop a software system that took the values of the metrics and carried out their aggregation, thereby obtaining quality values for the high-level elements in the model. This system would consist of three parts: a "core" that performed the aggregations and stored the results of the evaluation in a database, a Maven plugin that allowed the automated execution of the "core", together with a web tool that showed the results of the evaluation in a helpful, attractive and practical way.
- Defining, documenting and establishing the Quality Management System of the laboratory. The QMS would consist of a Quality Manual, along with

administration and technical processes, technical instructions, records and formats. For example, some of the administration processes involved internal auditing, personnel training and qualification, documentation control, control of non-conformities, corrective and preventive actions, and management reviews. Some examples of the technical processes carried out are: result quality assurance, validation of the quality model, validation of the software analysis tools used in evaluations, report elaboration and test item manipulation. Some examples of technical instructions are the ones that define how to configure the execution of the different analysis tools in the context of the evaluation framework, as well as their installation and deployment.

Seeing that the scope of the matter at hand was quite vast and our team size was very small (three people), we saw fit to use an iterative and incremental approach. In addition, the scope of some of the tasks was uncertain initially, and we knew some of the requirements might change during the implementation; (for example, when we started the definition of our quality model and evaluation process, the ISO/IEC 25010 and ISO/IEC 25040 standards were still draft versions that might have changed once the final version was released).

Even though Scrum was developed with software development in mind, we found similarities between software development and what we had to do, as both are unique creative efforts that require the development of different components and demand knowledge in diverse areas. Nevertheless, our endeavor also entailed software development, since we had to develop the evaluation framework. For that task we were also able to take full advantage of Scrum.

All of these circumstances led us to choose Scrum as the best approach for managing the implementation of AQC Lab.

Of course, we knew there were also differences between implementing AQC Lab and developing software. For instance, the extent of research, experimentation and validation involved was larger in our case than what you typically have in a software development project. The particular circumstances of our context made it necessary to adapt some of the practices of Scrum, while at the same time ignoring some of the rules.

Below is a description of the adapted Scrum process that we followed:

- We kept the three roles described in Scrum. In our case there was no external client; because of that, the role of Product Owner was assigned to our CTO, as he had the appropriate characteristics: product vision and leadership. As we were a small team of just three people, the role of Scrum Master was shared by the same person as above, since he also had the best characteristics to fill this particular role; he possessed the capacity to facilitate the process, resolve impediments, enforce time boxes and promote improvement. The other two people made up the Development Team, although in the implementation of a few of the elements of the Product Backlog, the person holding the Product Owner/Scrum Master roles also took a small part in the Development Team. Having one person that plays both the roles of Scrum Master and Product Owner is considered among most practitioners to have potential for a conflict of interests, due to the fact that the same person is

responsible for supporting and protecting the team, as well as for "pushing" the team to get more business value out of the product being developed. Even thought the potential for conflict exists, it does not necessarily have to materialize if the person playing both roles finds the right balance between the interests related to each of them. In a very small team without external client, the dual role solution is perfectly viable as long as the Scrum Master/Product Owner is able to support the team while ensuring they keep a sustainable development pace. In this case, time constraints may be the main problem to deal with, as performing the tasks related to both roles can be quite time-consuming.

- We kept all of the elements to implement in the Product Backlog (PB), and made it accessible for everyone via Google Docs (now Google Drive). The items in the PB were prioritized by the Product Owner. The PB was a living artifact, dividing the top-priority items into more specific and granular ones when we had enough knowledge to do so. Occasionally, implementing some items led to the discovery of new requirements, as well as to a change of scope; this was subsequently reflected in the PB. The whole team took part in estimating the effort that PB items would take.
- The effort required for the elements in the PB was not always as small as recommended by Scrum experts. Due to some of the items requiring a lot of research and trial and error, it was impossible to break them down into smaller items. For this same reason, some of the items did not fit a sprint, which meant that we broke one of the rules of Scrum.
- We had a Sprint Backlog (SB) for each Sprint. An example of the structure of the SB is given in Table 1. The items in the Sprint Backlog were extracted from the top-priority items in the PB. We kept the status of each item ("pending" – "done") in the SB, along with an estimate of the remaining time to be completed. This estimate was updated by the Development Team every day after the Daily Scrum. We also used Sprint burn-down charts to monitor the progress of the Sprint and the remaining effort, since the SB always had the information of the remaining effort updated to present estimates (Fig. 2 shows an example). The SB, like the PB, was accessible via Google Docs. We did not find it necessary to have a physical board to keep the information about the tasks, as we found it more useful to keep it centralized in the SB.
- We started each Sprint with a Sprint planning meeting where the whole team took part. The first point in this meeting was to establish the duration of the Sprint. At first, the Sprints were three weeks long, but we later decided to make them four weeks long, since the nature of the tasks (longer than what is usual in software development) made it feel more consistent to have longer Sprints. The main advantage of a shorter sprint is allowing the team to detect earlier if the product being developed does not meet the needs of the client. This way, the risk of developing the wrong product is reduced. In our case, this potential risk was not a problem, since the Product Owner attended the Daily Scrums and was completely aware of the progress being made during the Sprint. More information about the Sprints is given in Table 2. Once the duration of the Sprint was established, the team revised the PB and decided as a group which set of PB items would be

implemented. The set was decided based on the priorities assigned by the Product Owner, as well as the effort estimation made by the whole team. Based on this information, the team would decide which set of items would be achievable in the time box established for the Sprint. Once the items were chosen, the whole team debated which tasks each item would entail and in which order they should be performed. The list of tasks to perform for each item was also included in the SB. The recommendation to have tasks that require one day or less of work was often difficult to fulfill, due to what has been explained above –the degree of research and experimentation involved. The planning meeting usually took us about an hour.

Table 1. Structure of the Sprint Backlog and example of part of its content during Sprint #9

PB Item	Remaining effort	Status	Tasks
Visualization module: AQC Lab-web	12	Pending	Pending: - Include Line chart for evolution of Characteristic values for selected project in Historic page - Include Line chart for evolution of Subharacteristic values for selected project in Historic page Done: - Include TreeMap chart with info from all projects in Home Page - Include Kiviat chart of selected project in Characteristic page - Include Kiviat chart of selected project in Subcharacteristic page …
Validation of the test method	0	Done	Pending: Done: - Research and select software products to use in validation - Download source code of selected products - Evaluate selected products - Extract information from bug tracking systems
…	…	…	…

- In each Sprint, the team performed the different tasks for the PB items that had been committed to in a collaborative way. For the PB items that involved software development, the recommendations of Scrum were followed: the team focused on producing, within the Sprint, software that had been tested and which, at the end of the Sprint, actually worked. For other PB items, like documents or forms, we also tried to always have a revised version at the end of the Sprint.

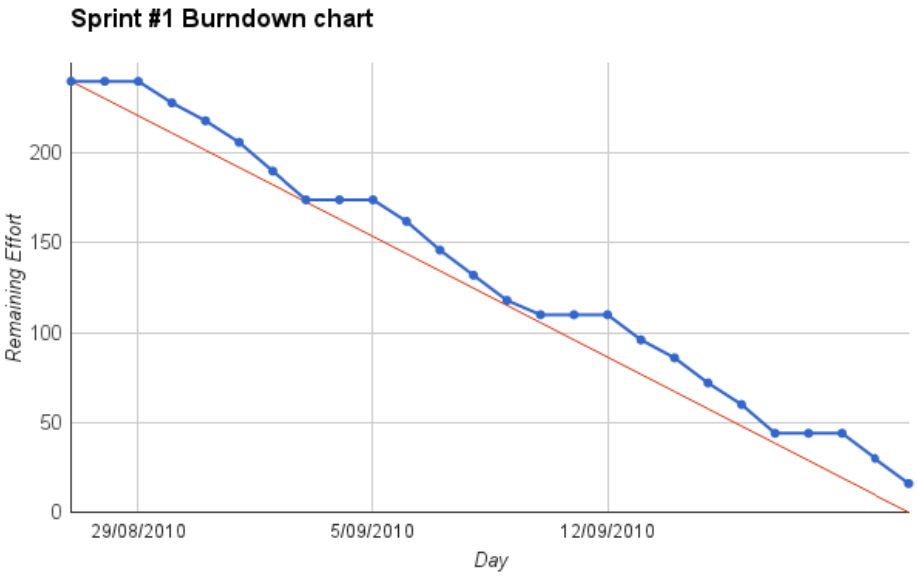

Fig. 1. Burn-down chart for Sprint #1. The line with dots shows the estimated remaining effort, which was updated after the Daily Scrum. The straight thin line represents the ideal trend. As we can see in this figure, the initial estimations were too optimistic and the team could not keep the velocity necessary to complete all the committed PB items.

- We conducted a Sprint review and retrospective meeting after every Sprint. We usually had the review meeting first, which took us about an hour; right afterwards, we had the planning meeting for the next Sprint. The whole team took part in the review, and the Product Owner led the discussion about which committed PB items had been done and which had not been carried out. The experimental nature of some of the tasks made them rather unpredictable as regards the effort they would take. Because of that, in the first Sprints our estimates were quite off target and usually there would be many unfinished items at the end of each Sprint. Nonetheless, as we gained experience, our estimates of the effort required for the PB items and related tasks got better, which led us to choose a more adequate amount of tasks to perform in later Sprints. After reviewing the work completed, the group then discussed what problems they had had during the Sprint, outlining how they were solved. We then had a live demonstration of the work products that involved software development (some elements of the evaluation framework). For other work products, we did not usually have a live demonstration of what had been done during the Sprint, because it was often the case that there was not a product to show *per se*. The result of a task would often be a document or a section of a document, and these results were reviewed as they were produced, not at the end of the Sprint. After the review, we would usually have a quick retrospective meeting to look back on and discuss the process. It usually took only a short time, as the whole team was comfortable with the process and we all felt few adjustments were necessary.

Table 2. Details of some Sprints in the implementation of AQC Lab

Sprint #	Dates	Main goals
01	27/08/10 - 21/09/10	Research and define Metrics for Maintainability. Research tools that provide values for the metrics in the Maintainability model.
02	21/09/10 – 15/10/10	Define functions to aggregate metric values and obtain values for high-level elements of the Maintainability model. Set thresholds for metric values.
03	18/10/10 – 12/11/10	Design the architecture of the evaluation framework. Document the Maintainability model. Produce the QMS Quality Manual. Refine metrics (filter rules from static source code analyzers).
04	15/11/10 – 03/12/10	Develop business domain and data layer of the evaluation framework. Document administrative procedures (documentation control, organization, and personnel). Document technical procedures (tool configuration).
05	03/12/10 – 24/12/10	Develop the evaluation engine of the evaluation framework (tool result integration and aggregation of values). Create personnel records (training, authorizations, etc.).
06	10/01/11 – 31/01/11	Develop the evaluation engine of the evaluation framework (tool result integration and aggregation of values). Define and document evaluation process (test method). Document administrative procedures (internal audits, management reviews).
07	31/01/11 – 18/02/11	Develop automation module of the evaluation framework (plugin for Maven). Document technical procedures (threshold revision). Create forms and records related to administrative procedures (document control, internal audits, etc.)
08	21/02/11 – 11/03/11	Develop visualization module of the evaluation framework (data visualization). Improve core of evaluation framework: improve multi-module product evaluation. Define procedure for validation of the test method.
…	…	…
22	02/05/12 – 01/06/12	Perform the internal audit. Carry out evaluation of product AAA for client BBB.
23	04/06/12 – 06/07/12	Define and implement corrective actions for non-conformities detected in internal audit. Carry out evaluation of product XXX for client YYY.
24	16/07/12 – 10/08/12	Receive accreditation audit. Produce documentation requested by accreditation body auditors. Define and implement corrective actions for non-conformities detected in accreditation audit.

- We conducted Daily Scrums; the whole team, including the Product Owner, took part in these. In these quick meetings each team member reported on what had been done the previous day, the problems he had faced, and what he would do that day. The Daily Scrums were usually kept to no more than fifteen minutes, though there were days on which discussing some topics (like how to tackle the problems the team members faced) would prolong the meeting. However, we sometimes found it useful to go beyond the fifteen minute time-box, since this gave us a good opportunity to share the vision of the Product Owner about topics that mattered to the development team.

4 Conclusions and Future Work

Methodologies and process frameworks, such as Scrum, are supported by a lot of effort and empirical research whose goal is to test how the practices they define interrelate and work together to attain their intended benefits. Each one of their components and practices serves a specific purpose and is essential to the successful usage of the methodology or process framework. In a nutshell, they are not part of the methodology simply because of some whim; teams have to take that into account when they adapt a methodology to their own circumstances.

Nonetheless, we advocate for a contextual approach to Agile methods, adapting the elements to suit the context in which they are adopted. It seems paradoxical to affirm that an Agile method cannot be adapted and that it must be followed strictly.

Even though it is a software development framework, we found Scrum really useful for our purpose of implementing a software evaluation laboratory. We believe, moreover, that it can be easily adapted to other contexts outside the realm of software development, since it focuses mainly on project management. We do concur that method adaptations have to be done carefully, though. You have to be perfectly clear about the purpose of each element of the method that you are changing, as well as how that change affects what really matters, i.e., the performance of the team.

We found Scrum to be really well-crafted for project management. It enabled us to get different levels of zoom on the information required to monitor the progress of the project:

- The Product Backlog provided us with a general snapshot of what has to be done, with the advantage that it was not a static snapshot, since it was updated constantly to reflect newly-discovered scope throughout the project. Moreover, this snapshot provided closer detail about what was most important at each point in time, via the priorities specified for its items.
- The Sprint Backlog provided a sharper focus on what was important within the time box of a month (or less). It allowed us to concentrate on the most urgent items, taking an incremental approach that made implementation easier.
- The Daily Scrum allowed us to know how we were progressing on a day-by-day basis. We found the Daily Scrum to be the most useful practice in Scrum, as it improved our decision-making by keeping the whole team involved. It made for

better performance; since everybody in the team knew what the others were doing, each individual could lend a hand to other members when they had issues to solve.

Achieving the accreditation may be considered a major milestone in the implementation of AQC Lab. However, that did not mean that we had reached the end of the road. Since the accreditation, we have been working on expanding the scope of the evaluations carried out by the laboratory; we are defining evaluation models for other quality characteristics, like Functional Suitability and Usability, or researching and developing tools to measure the metrics related to those characteristics; we are also researching tools to evaluate Maintainability on software products developed with other programming languages not supported initially, like Groovy and Objective C, etc. We are still using Scrum to manage all this work.

In addition, maintaining accreditation requires the continual improvement of the QMS that governs the activity of the laboratory. This correct operation and improvement is monitored by the accreditation body via follow-up audits. We have just received our first follow-up audit and we have had very positive feedback from the audit team. The effectiveness of our QMS is a consequence of the fact that we also use Scrum to manage the operation of AQC Lab, which involves carrying out software quality evaluation tests and performing administrative and technical activities, as well as implementing improvement actions.

Acknowledgements. This work has been funded by the GEODAS-BC project (Ministerio de Economía y Competitividad and Fondo Europeo de Desarrollo Regional FEDER, TIN2012-37493-C03-01) and by the ECU: Evaluación y Certificación de la fUncionalidad del Producto Software project (Consejería de Empleo y Economía y Fondo Europeo de Desarrollo Regional FEDER, 1313CALT0056).

References

1. SO/IEC 25000:2005 - Software Engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Guide to SQuaRE. International Organization for Standardization, Geneva, Switzerland (2005)
2. ISO/IEC 25010:2011 - Software Engineering – Software product Quality Requirements and Evaluation (SQuaRE) – System and software quality models. International Organization for Standardization, Geneva, Switzerland (2005)
3. ISO/IEC 25040:2011 - Software Engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Evaluation process. International Organization for Standardization, Geneva, Switzerland (2005)
4. Schwaber, K.: Scrum Development Process. In: Business Object Design and Implementation, pp. 117–134. Springer, London (1997)
5. ISO/IEC 17025:2005 - General requirements for the competence of testing and calibration laboratories. International Organization for Standardization, Geneva, Switzerland (2005)
6. Beck, K.: Extreme Programming Explained: Embrace Change. Addison-Wesley, Boston (2000)

7. Hoda, R., Kruchten, P., Noble, J., Marshall, S.: Agility in Context. In: Proceedings of the ACM International Conference on Object Oriented Programming Systems, Languages, and Applications (OOPSLA 2010), pp. 74–88. ACM, New York (2010)
8. Conboy, K., Fitzgerald, B.: The Views of Experts on the Current State of Agile Method Tailoring. In: McMaster, T., Wastell, D., Ferneley, E., DeGross, J.I. (eds.) Organizational Dynamics of Technology-Based Innovation: Diversifying the Research Agenda. IFIP International Federation for Information Processing, vol. 235, pp. 217–234. Springer, Heidelberg (2007)
9. Hicks, M., Foster, J.S.: Adapting Scrum to Managing a Research Group. Technical Report CS-TR-4966, University of Maryland, Department of Computer Science (2010)
10. Willeke, M.H.H.: Agile in Academics: Applying Agile to Instructional Design. In: Proceedings of the 2011 Agile Conference (AGILE 2011), pp. 246–251. IEEE Computer Society, Washington (2011)
11. van Solingen, R., Sutherland, J., de Waard, D.: Scrum in Sales: How to Improve Account Management and Sales Processes. In: Proceedings of the 2011 Agile Conference (AGILE 2011), pp. 284–288. IEEE Computer Society, Washington (2011)
12. DeFauw, R.: Can Marketing Go Agile? In: Proceedings of the 2012 Agile Conference (AGILE 2012), pp. 136–140. IEEE Computer Society (2012)
13. Mazzanti, G.: Agile in the Bathtub: Developing and Producing Bathtubs the Agile Way. In: Proceedings of the 2012 Agile Conference (AGILE 2012), pp. 197–203. IEEE Computer Society (2012)